Informatics

# Efficient Reuse and Variability Management for Cyber-Physical Production System Families

DISSERTATION

zur Erlangung des akademischen Grades

**Doktor der Technischen Wissenschaften**

eingereicht von

**Dipl.-Ing. Kristof Meixner, BSc.**
Matrikelnummer 09725208

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.Ing. Mag. Dr. Stefan Biffl
Zweitbetreuung: Univ. Prof. Mag. Dr. Rick Rabiser

Diese Dissertation haben begutachtet:

Michael Felderer

Leopoldo Motta Teixeira

Wien, 8. November 2024

Kristof Meixner

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

# Efficient Reuse and Variability Management for Cyber-Physical Production System Families

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

**Doktor der Technischen Wissenschaften**

by

**Dipl.-Ing. Kristof Meixner, BSc.**
Registration Number 09725208

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.Ing. Mag. Dr. Stefan Biffl
Second advisor: Univ. Prof. Mag. Dr. Rick Rabiser

The dissertation has been reviewed by:

| Michael Felderer | Leopoldo Motta Teixeira |
|---|---|

Vienna, November 8, 2024

Kristof Meixner

# Declaration of Authorship

Dipl.-Ing. Kristof Meixner, BSc.

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.
I further declare that I have used generative AI tools only as an aid, and that my own intellectual and creative efforts predominate in this work. In the appendix "Overview of Generative AI Tools Used" I have listed all generative AI tools that were used in the creation of this work, and indicated where in the work they were used. If whole passages of text were used without substantial changes, I have indicated the input (prompts) I formulated and the IT application used with its product name and version number/date.

Vienna, November 8, 2024

Kristof Meixner

# Acknowledgements

The endeavor of my PhD would not have been possible without the help of many people. I would like to thank my supervisor, Stefan Biffl, for his guidance and support throughout my PhD. Thanks for the extensive discussions, countless influential ideas, and continuous feedback. I am also grateful to my supervisor, Rick Rabiser, for his thoughtful advice and different perspectives that enriched my research. I would like to extend my thanks to Michael Felderer and Leopoldo Motta Teixeira for agreeing to review my thesis. Thanks also go to Stefan Woltran and Wesley Assunção for being part of my proficiency evaluation and defense committee.

Special thanks go to my colleagues at CDL-SQI. I would particularly like to highlight and express my gratitude to Felix Rinker for the many late-night discussions, feedback on numerous publications, in particular, on this thesis, and fun times over the years. Thanks should also go to Jakob Decker and Hannes Marcher for working together on various prototypes. Many thanks to Dietmar Winkler, especially at the beginning of my thesis. I also would like to thank numerous students that I was able to support during their theses and with whom I experimented on different approaches and prototypes.

Most of the research presented in this thesis was conducted in collaboration with outstanding partners. I want to thank all my co-authors for their great cooperation and dedicated work, which often made my work fun. Particularly, I would like to thank Kevin Feichtinger for many fruitful discussions that sparked new ideas and many evenings crunched over publications that had to be written. Also, thanks to the co-organizers with whom I had the gratitude to organize several workshops and special sessions. I would like to extend my appreciation (alphabetically) to Aljosha, Alois, Arndt, Bianca, David, Laura, Lisa, Matthias, Paula, Sayyid, Fabian, Siwara, and Virendra. Many thanks to the many others I am not mentioning here by name but whom I am thinking of.

I want to express my profound gratitude to my family, mother, father, loving sister, daring brother, and partner for their endless support over time and throughout some of the most challenging times in life. This accomplishment would not have been possible without every single one of them. In loving memory, I am incredibly thankful for Cookie, who acted as our office dog and cheered up so many of the research community when accompanying me on several occasions.

My sincere gratitude and love go to my partner, Julia Obdrzalek, for her unwavering patience and support. Completing tasks often demanded a significant portion of our

time together, requiring sacrifices I deeply appreciate. I am profoundly grateful for her being just there and having an open ear for when I needed to talk and moments of doubt, always providing comfort and reassurance.

Finally, I would like to thank all my friends for their longing friendship, even in times when we had few opportunities to meet or talk.

# Abstract

Today, customers demand highly individualized products, such as cars, significantly increasing the number of product variants, which drives flexible and reconfigurable manufacturing. Cyber-Physical Production Systems (CPPSs), such as automated car factories, are software-intensive systems that use modern manufacturing techniques to produce such customizable products. They contain production resources, such as robots, that execute production processes like welding car parts. During CPPS engineering, domain experts aim to systematically and efficiently reuse engineering artifacts. Thereby, they built CPPS families with common and variable features, such as work lines with similar production resources.

However, CPPS engineering involves multiple disciplines, such as mechanical and electrical engineering. This setting creates challenges, including discipline-specific implicit CPPS engineering knowledge and a mainly manual engineering process that is inefficient and hard to reproduce. At the same time, separating the discipline-specific concerns is crucial for agile CPPS engineering.

This cumulative thesis aims to address these challenges through a multifaceted approach. First, it aims to establish explicit *production knowledge models* to facilitate multidisciplinary CPPS knowledge modeling and exchange. Second, it investigates designing methods, reference models, and design patterns for *advanced CPPS engineering applications* to improve knowledge communication and coordination. Finally, it aims to develop *integrated reuse and variability management* to enhance the efficiency of CPPS family design.

Therefore, the thesis presents the following contributions. In the context of *production knowledge models*, we present the superimposed Product-Process-Resource (PPR) model that serves as a visual representation of PPR with variability. Then, we discuss the PPR meta-model as a formal model-based representation with variability and the Product-Process-Resource Domain-Specific Language (PPR–DSL) to make implicit engineering knowledge explicit. For *advanced CPPS engineering applications*, we present the Industry 4.0 Asset Network (I4AN) meta-model and the I4AN reference model for improved coordination and communication of engineering knowledge. Furthermore, we introduce basic engineering design patterns to address common engineering problems. Additionally, we present the Capability and Skill Reuse (CSR) framework to reuse engineering knowledge systematically. In the context of *integrated reuse and variability management for CPPS* *engineering*, we introduce the systematic, semi-automated, and integrated Extended Iterative Process Sequence Exploration (EIPSE) approach. EIPSE aims to manage the three primary dimensions of variability that PPRs spawn in CPPS families.

We evaluated the approaches with a mixed-methods approach (domain analyses, iterative internal validation, feasibility studies, case studies, user study) with engineers and domain experts from the CPPS domain and our industry partners.

The findings showed that the *production knowledge models* enable the straightforward externalization of engineering knowledge with variability. The *advanced CPPS engineering applications* demonstrated an improved multidisciplinary knowledge communication in industrial practice and increased reuse of engineering artifacts. The EIPSE allows an efficient represent CPPS engineering knowledge in verifiable variability models. Furthermore, evaluating the EIPSE with domain experts suggests that the approach allows the efficient configuration of the three primary dimensions of CPPS variability. As a result, the approach allows the reuse of engineering knowledge and artifacts and enables an efficient management of their variability. These findings, we argue, indicate that the approach can reduce the effort required from domain experts when evolving or designing CPPS families.

# Contents

**Abstract** **ix**

**Contents** **xi**

**1 Introduction** **1**
1.1 Problem Definition . . . 5
1.2 Related Work . . . 15
1.3 Research Goals . . . 23
1.4 Research Methodology . . . 24
1.5 Outline . . . 31

**2 Synopsis of Publications** **33**
2.1 Production Knowledge Models . . . 33
2.2 Advanced CPPS Engineering Applications . . . 37
2.3 Integrated Reuse and Variability Management for CPPS Engineering . 38

**3 Scientific Contributions** **43**
3.1 Production Knowledge Models . . . 43
3.2 Advanced CPPS Engineering Applications . . . 53
3.3 Integrated Reuse and Variability Management for CPPS Engineering . 63

**4 Conclusion** **77**
4.1 Discussion . . . 77
4.2 Conclusion . . . 83

**5 Publications** **85**
5.1 Production Knowledge Models . . . 85
5.2 Advanced CPPS Engineering Applications . . . 115
5.3 Integrated Reuse and Variability Management for CPPS Engineering . 146
5.4 Additional Publications . . . 202

**A Overview of Generative AI Tools Used** **207**

**List of Figures** **209**

**List of Tables** 211

**Acronyms** 213

**Bibliography** 217

CHAPTER 1

# Introduction

Today, customers demand highly individualized products, such as cars, electronics, or other consumer goods, driving the need for flexible manufacturing. This demand is related to rapid innovation and shorter product lifecycles that significantly increase the number of product variants [84]. For instance, Scania, a large Swedish producer of trucks and buses, allows their customers to configure around a thousand individual features of their products [147]. Figure 1.1 shows a simplified example to illustrate the concepts, following the Product-Process-Resource (PPR) paradigm [181], throughout the subsequent paragraphs. The leftmost column depicts five similar but different car types with various bodies, windows, and tires. The second column shows related production processes for manufacturing the car types. The third column depicts the production resources and their layouts on the shop floor. The rightmost column illustrates two related Cyber-Physical Production Systems (CPPSs) manufacturing the car series.

CPPSs are software-intensive systems that use cutting-edge Information and Communication Technology (ICT) to manufacture such highly configurable *products* from a product portfolio using modern manufacturing methods [69, 143]. CPPSs consist of *production resources*, such as robot arms and conveyor belts, managed by software, such as control software and data-intensive information systems, which interact with the physical environment via sensors and actuators [69, 143]. For instance, in automotive manufacturing, automated assembly lines assemble cars, often based on the configuration of consumers that choose particular features that the product "digitally carries" along the production process.

Hence, CPPSs need to execute a flexible series of *production processes*, which build the glue between products and production resources, to manufacture the different product types [85, 143]. The second column of Figure 1.1 shows an exemplarily configured sequence of production processes. These car types build a *family of products* with the same platform but with different features. The first three car types share the same assembly process of the chassis (*C1*), while the fourth and fifth cars have slightly different

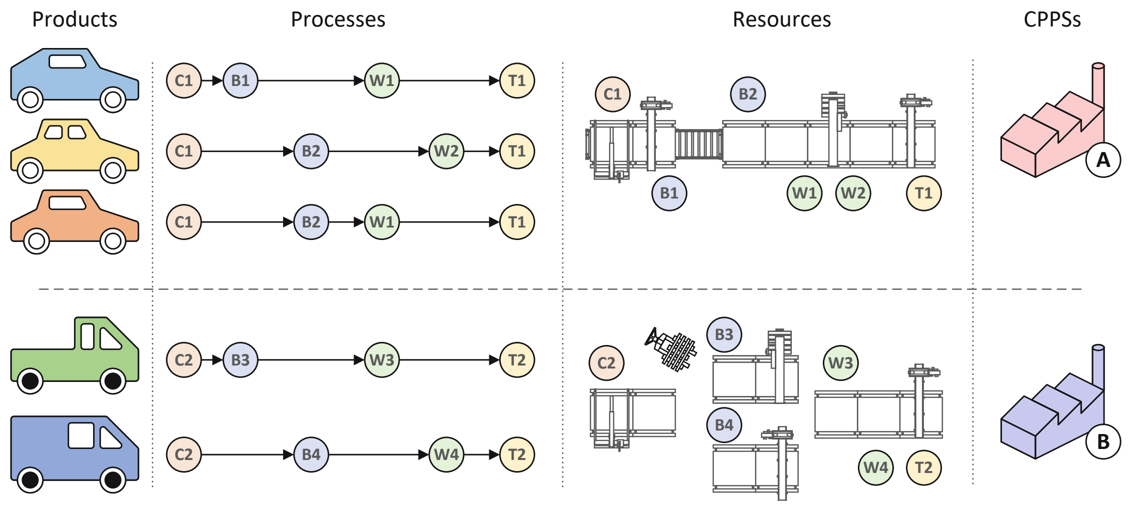


Figure 1.1: Illustrative car production example with its four dimensions of variability.

chassis assembly processes. Mounting the back is similar for the second and third car (*B2*), in contrast to mounting the back of the first car (*B1*) and the other vehicles (*B3* and *B4*). Similarly, assembling the windows is the same for the first and third car (*W1*) and different but related for the other car types (*W2*, *W3*, and *W4*). In the final production process step, the tires are mounted the same way for the three first and for the fourth and fifth car types.

Product Family.

A product family shares a common platform but has, at the same time, distinct features [19, 152]. For instance, a car series typically shares a common platform with various individual features that customers can configure.

Per product type, these production processes are executed, with some degree of freedom, in a particular sequence by production resources. Figure 1.1 shows such production resources in the third column as two work lines. For instance, assembling the chassis and mounting the tires is done on similar production resources (*C1*, *C2* and *T1*, *T2*) with different configurations. Mounting the back onto the cars is done sequentially on two different production resources for the first three cars (*B1* and *B2*) and in parallel in the case of the flatbed and small truck (*B3* and *B4*). This differentiation of the production resources is similar to installing the windows, which is done on different production resources (*W1*–*W4*).

Consequently, CPPSs require capabilities for extensive and more efficient flexibility in reconfiguration (brownfield engineering) and new design scenarios (greenfield engineering) [85, 154]. Envisioned by the Industrie 4.0 initiative [202],[1] the interplay of knowledge,

[1] Researchagenda Industrie 4.0: `https://www.bmbf.de/bmbf/de/forschung/`

software, hardware and environment aims to equip CPPSs with such production capabilities. This way, CPPSs aim at fostering flexible production down to small batch sizes and single customizable products in production operation, i.e., *lot size 1* production [85]. This objective goes along with the rapid adaptation to changing requirements, such as in production capacity, dispatching of orders [84], or new product types. Going beyond, the objective concerns the vision of adaptation to uncertain environmental conditions such as temperature or light irradiation [2, 183].

While much of the work lines are similar in the example of Figure 1.1, the products are manufactured in two different CPPSs, for instance, for production efficiency. This way, the two systems build a *family of CPPSs* on their own as they share, for example, a particular architecture concerning the hardware and the software. These circumstances result in an increased variability of CPPS designs, directly representing the state of the practice. These organizations repeatedly have to consider different characteristics, such as plant setups, hardware, geolocations, and recent advances like frugal production [53].

CPPS Family.

A CPPS family shares a common platform, for instance, system architecture like step-chain work lines or workshop production with similar production resources, such as the same type of robots and their control software. However, the individual CPPSs are, for instance, planned with variations like manufacturing different product families in various geolocations.

Together, the *family of products*, the *family of production processes*, the *family of production resources*, and the *family of CPPSs* as a multi-product line [4, 78] build *four principal dimensions of CPPSs variability* [55].[2] Therefore, CPPS engineers reuse production engineering knowledge and artifacts, such as product and resource models, design components, or control code [71, 178]. Thereby, they often build CPPS families in an implicit or ad-hoc manner with the problem, for instance, to propagate changes to CPPS family instances. At the same time, knowledge and artifact reuse typically follow a *clone-and-own* approach [54]. In contrast, SPL engineering emphasizes systematic reuse, variability modeling, and management of (software) artifacts and models across products [152]. Hence, SPL engineering aims at proactively planning for reusable artifacts and systematically and efficiently reusing the artifacts by managed variability, i.e., modeling and managing the commonalities and the differences in the systems [152]. In this regard, the SPL research community has recently shown a growing interest in applying SPL concepts to CPPS engineering [1, 22, 104].

However, beyond SPL engineering, engineering a complex CPPS is an effort of engineers from multiple heterogeneous disciplines, such as mechanical, electrical, and software

`digitale-wirtschaft-und-gesellschaft/industrie-4-0/industrie-4-0_node.html`

[2] We use the term *product families* and *product lines* synonymously. However, we tried to use the term *product families* in the context of CPPSs and *product lines* in the context of Software Product Line (SPL) research.

engineering, with different views on the system [10, 198]. For instance, engineers of the so-called *basic planning phase*, i.e., conceptualization and rough system planning, draft a functional *CPPS design* [109, 198] (cf. also Section 1.1.2). This design is based on the *requirements of a client*, who wants to buy or operate the CPPS. Furthermore, the design is complemented with a cost estimate for the client. In case the client accepts the design and cost estimate, the engineers of the various involved disciplines pick up the system design drafts and refine them in the *detailed engineering phase* [109, 198]. One issue that challenges engineering organizations is that much of the knowledge used to design a CPPS and its processes is *implicit knowledge* of, mostly senior, domain engineers *scattered over different heterogeneous disciplines*. Subsequently, this makes the engineering process *hard to reproduce* which impedes flexible reconfiguration.

Similar to software systems, CPPSs undergo a regular evolution, which means that planning repetition may be necessary, for example, due to changes in product and customer requirements [195]. Such an evolution requires remodeling, reimplementing, and revalidating the CPPS design throughout the involved engineering disciplines to incorporate the changes [198], which adds to the complexity. If, for example, parts of the car platform change, the compatibility of all car types and CPPS designs must be re-checked for potential issues and subsequent redesign tasks.

CPPS Reuse and Variability.

Establishing CPPS reuse and variability involves proactively planning for reusable engineering knowledge and common CPPS platforms (based on [152]). This activity aims to establish reusable engineering artifacts with their dependencies, such as production process or resource specifications, from multiple disciplines and viewpoints. Engineers must model and manage the reusable knowledge and engineering artifacts with their variability to achieve systematic and efficient reuse of engineering knowledge and assets. These efforts include reducing domain expert effort for/with reuse or increasing the scope and value of reused engineering assets.

Nevertheless, engineers aim to reuse and share existing concepts and components, to improve engineering quality and efficiency while lowering engineering and maintenance effort and risk [71, 178]. This reuse concerns, for instance, using the same types of robots where possible. These design artifacts stem, for instance, from prior projects in a similar engineering discipline. Therefore, the engineers and disciplines share engineering artifacts that represent *CPPS assets*, i.e., material or immaterial objects [160, 200]. Beyond those artifacts, engineers require accurate *knowledge models* of CPPS concepts with variability to effectively and efficiently fulfill their engineering tasks and collaborate across engineering processes [7, 200, 201]. Practitioners also disclosed that *applications for CPPS engineering* are often insufficient for systematic CPPS knowledge representation, mainly concerning structured knowledge reuse. This issue further calls for approaches and techniques for *efficient and integrated reuse and variability management* in CPPS engineering.

To tackle challenges of security and production quality improvement for CPPSs, the Christian Doppler Laboratory For Security and Quality Improvement in the Production System Lifecycle (CDL SQI) was established.[3] This thesis and its topics are embedded in the CDL SQI, engaging in foundational research backed by essential research challenges of industry partners. Furthermore, it aims to foster collaboration between local and international researchers and PhD students. Throughout this thesis, the author of this thesis collaborated closely with many researchers and practitioners. The publications contributing to this cumulative thesis are all 1st author publications apart from one 2nd author publication (cf. Table 2.1). Other contributions in further publications can be clearly distinguished from those of other PhD students and researchers by marking the author's position (cf. Table 5.1).[4]

This thesis is based on the assumption that achieving the required CPPS flexibility and adaptability and overcoming the impediments for CPPS engineering requires integrating approaches and techniques from multiple research fields. We argue that a suitable integration requires new methods and an interdisciplinary research effort, bringing together research from the fields of *CPPS Engineering*, *SPL Engineering*, and *Model-based Software Engineering (MBSE)* [117]. This thesis aims to develop methodologies and tools that effectively address the intricacies of variability management within CPPSs engineering by intertwining contributions coming from these research disciplines to establish the following research lines as main themes:

**RL1**. Introducing *Production Knowledge Models (PKMO)* shall make implicit production engineering knowledge with variability explicit and mitigate issues coming from heterogeneous multidisciplinary engineering artifacts as a foundation for

**RL2**. *Advanced CPPS Engineering Applications (ACEA)* that shall provide the foundations for engineers to reuse multidisciplinary production engineering knowledge systematically and facilitate

**RL3**. *Integrated Reuse and Variability Management (IRVM) for CPPS engineering* to manage the variability of CPPS families, particularly considering production process sequences to facilitate efficient reuse in comparison to a traditional manual approach.

## 1.1 Problem Definition

Engineering CPPS is an extensive effort of multiple stakeholders that come from different, heterogeneous disciplines. Improving this engineering process for efficient reuse and variability requires suitable methods and techniques. There exist frameworks for introducing reuse in the production systems domain, such as the VDI 3695 guideline [200], that resembles *Domain and Application Engineering (DAE)* from SPL engineering [152]

[3]CDL SQI: `https://sqi.at`

[4]Disclaimer: This thesis uses the *we* form rather than the *I* form throughout the text. This is owed to numerous discussions with colleagues and other researchers, for instance, at conferences, all contributing at least partially to achieving the research results and outcomes.

(cf. Section 1.2). However, similar to the latter, the VDI 3695 guideline leaves concrete implementation open but proposes higher-level directives. For CPPS engineering, we argue it requires approaches for explicit knowledge management and its applications for CPPS engineering. Furthermore, it mandates an integrated management of the aspects of reuse and variability for multidisciplinary CPPS engineering.

This section first describes two illustrative use cases from an industry partner to frame the topic further and break it down into concrete research challenges. Secondly, the section describes the traditional CPPS engineering process with its issues as a baseline. Finally, this section raises practical engineering challenges that reflect the business needs of engineering organizations.

### 1.1.1 Industrial Use Cases

To illustrate the challenges concerning reuse and variability management in CPPS engineering, this section introduces two industrial use cases from an industry partner in the CPPS domain, the *shift fork* and the *rocker switch* [130]. These use cases were published, together with two other use cases, in the *Extractive Software Product Line Adoption (ESPLA)* case study catalog [115][5] as part of this thesis [130].

#### Shift Fork

The *shift fork* use case comprises a CPPS designed and operated by an industry partner. The CPPS manufactures four different types of shift forks [130]. Figure 1.2 depicts a simplified schematic drawing of two of the four shift forks with their parts. A shift fork moves a cuff with its fork, which consists of several fork jaws, along a pipe to engage and align the transmission gears. Typically, one shift fork positions one particular cuff for two different gears, such as the first gear and the third gear.

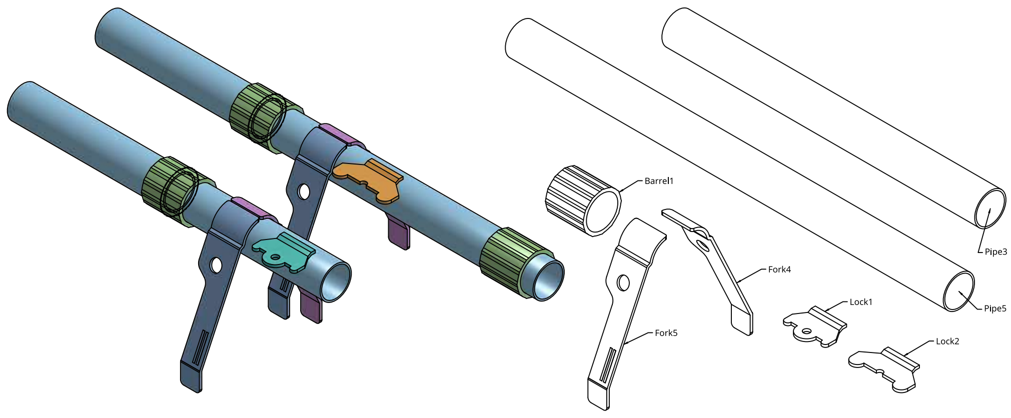


Figure 1.2: Rendered *shift forks* for a manual transmission with a list of parts.

[5]ESPLA catalog – `https://but4reuse.github.io/espla_catalog/ESPLACatalog.html`

The shift forks vary depending on the different gear combinations in the transmission. A shift fork consists of a *Pipe*, several *Forks*, a *Lock*, and several auxiliary parts, such as the *Barrels*. In this use case, the CPPS produces a product family of **4** distinct shift fork types for a 7-gear truck transmission constructed from **15** parts. Selective production steps must follow a specific sequence to produce the shift fork types. For example, the *Pipe* must be available in the production system before the *Forks* are mounted onto the pipe. However, several degrees of freedom in assembling a shift fork remain. For instance, it is not important whether the *Fork* on the left or right side are welded onto the *Pipe* first. The sequence of these production steps must be resolved during CPPS engineering or operation. Overall, the CPPS executes more than **19** production process steps, executed by **21** work cells that contain production resources, such as robot arms.

### Rocker Switch

The *rocker switch* use case comprises a complex CPPS that is designed by an industry partner and shall manufacture **12** different types of rocker switches [130]. Rocker switches are everyday devices for controlling electrical appliances, such as lights and window blinds. Figure 1.3 shows a simplified schematic view of the core of a rocker switch in vertical and horizontal views.

The rocker switches vary depending on how many functions they control, for instance, one or two lights with a single switch. Typically, the core of a rocker switch consists of a *Socket*, in gray, several contacts (*Pole* in orange, *Neutral* in salmon, *Changeover* in magenta, and *Off* in blue), one or more *Rockers*, in green that open and close the electric circuits, and several auxiliary parts. In this use case, the CPPS shall produce a product family of **12** rocker switch types for different applications. These rocker switches are constructed from up to **40** parts that can vary quite much depending on the type of rocker switch. Like the *shift fork* use case, selected production steps must follow specific sequences. For instance, Figure 1.3 reveals that the *Rocker* must be pressed into the *socket* before the *Off* contact but after the *Pole* contact to switch between them. However, for instance, mounting the *Screws* can be done at many points in the production process. Similar to the *shift fork* use case, this induces several degrees of freedom concerning the order of production steps. Overall, the CPPS shall employ more than **75** production process steps executed by **49** work cells.

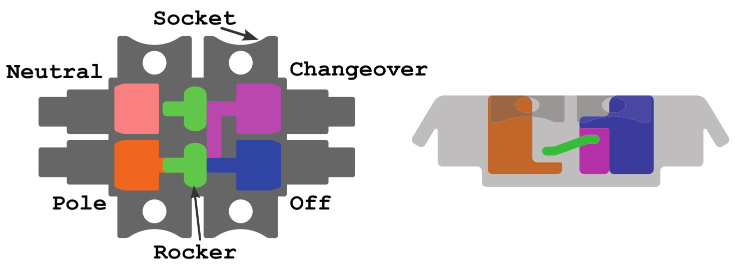


Figure 1.3: Vertical and horizontal views of a *rocker switch* and its components [130].

**Summary: Use Cases.** These two use cases illustrate typical product families manufactured on CPPSs. The main difference between the CPPSs is that the rocker switch core can have from **4** to **20** parts produced on **21** respectively **49** work cells, which increases complexity and potential configuration space by magnitudes. This problem complexity, especially, concerns the configuration sequence of production steps, **19** for the shift fork and up to **75** for the rocker switch, for optimal production regarding production resource utilization, availability, throughput, and initial and running cost. Furthermore, the solution complexity concerns investigating which rocker switch types are economically feasible and optimal under process and resource unit cost assumptions.

Sequence Configurations.

Configuring sequences of options results in the sequence roughly following a permutation with $P(n, r) = n!/(n-r)!$, with $n = r$ if the whole sequence is considered. This defines an even larger configuration space than independent optional binary features. For instance, the configuration space for 5 independent optional features would make $2^5 = 32$ configurations, but $5! = 120$ configurations considering the sequence.

### 1.1.2 Traditional CPPS Planning Process

Figure 1.4 shows the workflow of the traditional basic CPPS planning process elicited from an industry partner [87]. During this initial planning phase, the client provides the necessary product specifications and requirements towards the CPPS. These can be artifacts like Computer-Aided Design (CAD) drawings or prototypes of products with their parts or business constraints, such as the built-up area or cost.

#### Family of Products

The products comprise different types that usually form a (mechanical) product family. Designated engineers, known as *basic engineers*, analyze the product types and their commonalities and differences. This way, they determine a base type or shared platform on which the other product types shall build (Step 1). Based on this analysis, the engineers create a *bill of materials* that lists the parts required to manufacture each product variant.

One way to represent these lists for all product types of a family are Type Comparison Matrices (TCMs), a common approach in the industry that contain the product types in the columns and their parts in the rows. TCMs are easy to start with and manipulate in various spreadsheet tools, independent from the platform, and easy to understand when small. Table 1.1 shows the TCM of the shift fork product family with the shift fork types in the columns and their parts in the rows. For instance, *Fork-13* requires several parts but *Pipe 8* and *Lock 3* exclusively compared to the other shift forks. Further, *Barrel 1* is used in all shift fork types, while *Barrel 2* is only used in *Fork-13* and *Fork-57*.

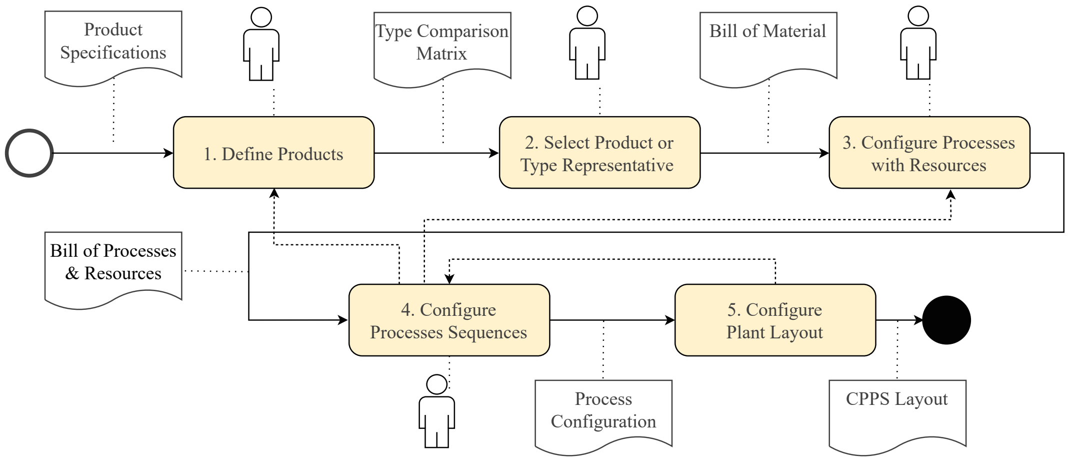

Product Specifications
Type Comparison Matrix
Bill of Material
1. Define Products
2. Select Product or Type Representative
3. Configure Processes with Resources
Bill of Processes & Resources
4. Configure Processes Sequences
5. Configure Plant Layout
Process Configuration
CPPS Layout

Figure 1.4: Traditional *basic planning* process of the initial CPPS design [87].

| Parts/Types | Fork-13 | Fork-2R | Fork-46 | Fork-57 |
|---|---|---|---|---|
| Pipe 2 | | | | × |
| Pipe 3 | | × | × | |
| Pipe 8 | × | | | |
| Barrel 1 | × | × | × | × |
| Barrel 2 | × | | | × |
| Jack 1 | × | × | × | × |
| Ring 1 | × | × | × | × |
| Fork 3 | × | × | × | × |
| Fork 4 | × | × | × | × |
| Fork 5 | × | × | × | × |
| Lock 1 | | | × | × |
| Lock 2 | | × | | |
| Lock 3 | × | | | |
| Screw | × | × | × | × |
| O-Ring | × | × | × | × |

Table 1.1: TCM of *shift fork* types (columns) and their parts (rows).

This way, our industry partner is already aware of the *structural product variability* and uses one kind of variability representation for the products. However, as our project partner confirmed, they get increasingly difficult to comprehend when extensive and error-prone to maintain. For instance, for this thesis, we elicited data from several documents and spreadsheets that engineers currently use with matrices up to 300 rows × 45 columns. Even worse, these matrices lack clear semantics within and across projects, making them hard to explain, even for the engineers. For instance, the original TCM for the shift fork never listed *Barrel 2*, but something that was assumed that the engineers know from the CAD drawings. Similarly, while the TCM for the shift fork contained × symbols to represent the presence of a part, the TCM contained integer values. Additionally, engineers often *clone-and-own* spreadsheets from one project to another, carrying over errors and semantic inconsistencies. This way, the spreadsheets are growing inconsistent over time, significantly challenging our project partner.

## Family of Production Processes

In the next step, the engineers examine plausible production process steps to assemble the product types, often by (virtually) disassembling the prototypes and putting them "back together" [105]. For instance, several process steps in the *shift fork* use case are joining the forks with the pipe by welding. The engineers begin by describing the atomic process steps required to assemble parts of a particular product variant.

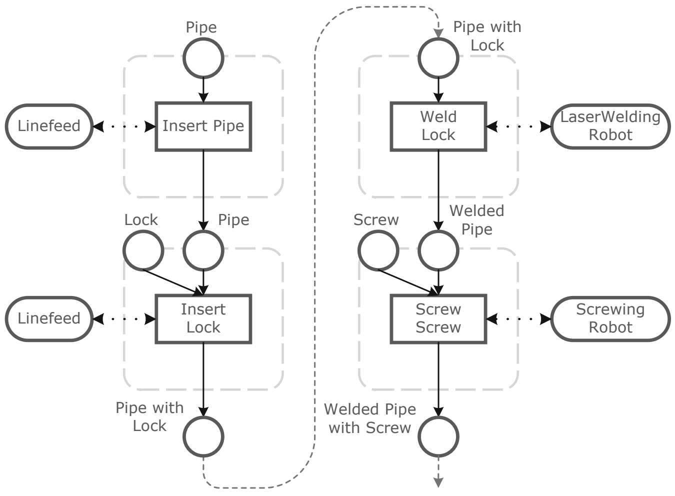


Figure 1.5: Partial PPR model for production of a shift fork in VDI 3682 notation [199] with *products* as circles, production *processes* as rectangles, and production *resources* as rounded rectangles, from Meixner et al. [137].

For planning this functional process perspective of the CPPS, the engineers can use representations [7], such as the VDI 3682 model and notation [199] that uses the concepts of the PPR paradigm [181]. For instance, Figure 1.5 shows a section of a production sequence for a single shift fork variant in a VDI 3682 model [199]. In the first step, a *Pipe* (product, depicted as a circle) *is inserted* (process, represented as a rectangle) into the CPPS using a *Linefeed* (resource, depicted as a rounded rectangle).

Combinations of these atomic process steps subsequently form composed production process steps and production process sequences, aiming for a feasible production process design. However, as mentioned in the prior paragraphs, several degrees of freedom remain when assembling a particular product variant. For instance, in the *shift fork* use case (cf. Section 1.1.1), the *Barrels*, the *Screw*, and the *Ring* can be assembled to the shift fork after welding the *Forks* in a very arbitrary order. This freedom results in a vast space of up to $n!$ possible production process variants (cf. Section 1.1.1). However, these options must be resolved in the engineering phase, for example, by engineers or during operation, for instance, by an information system that schedules the production process steps.

This task results in a *bill of processes* for each product variant. The engineers must merge these bills to represent one or more potential production process sequences for the requested product family. Take, for instance, the five production processes from the introductory example illustrated in Figure 1.1. Therefore, they often use a *type representative* that resembles a virtual product that combines several features of a set of product variants (Step 2). There, engineers have to investigate the commonalities and the variability of the processes and decide which process steps fit together and which sequences are feasible and efficient. This task is challenging itself as it presents, in its theory, a typical *set cover problem*, which is at least *NP-complete* if not *NP-hard* for optimizing production processes sequences.[6] Furthermore, it requires additional domain knowledge, for instance, which process steps are more prone to error, more resource intensive, or costly to define not a feasible but efficient process sequence. This is, among other things, why only experienced domain engineers with several years of experience are able to and trusted enough to take over the responsibility of making such wide-ranging design decisions.

Our industry partner has developed a closed-source, machine-interpretable data format for production processes. Process engineers also use this data format, for instance, to simulate the timings of production processes. Semantically it also seems beneficial to align such a representation with the taxonomies for production processes [31] and allow de-composition. However, so far, our industry partner has not developed a representation that incorporates the *behavioral process variability*.

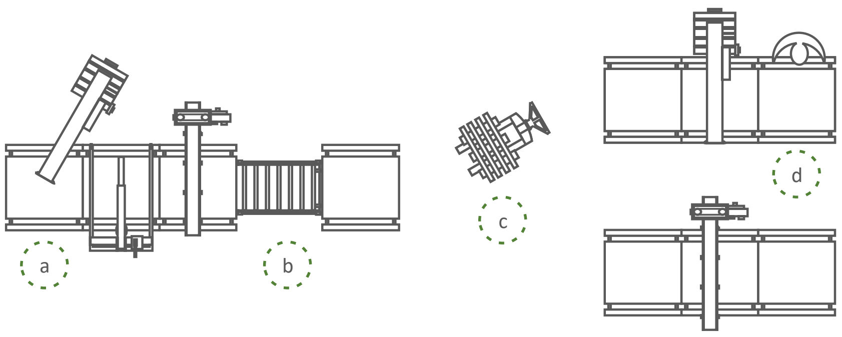


Figure 1.6: Illustrative section of a potential shop floor layout with production process labels.

### Family of Production Resources

To complete the first rough design of the CPPS, the engineers determine production resources, such as work cells or robot arms, resulting in a *bill of resources*. These

[6]Set cover problem: `https://w.wiki/Ai$f`

production resources must be suited to execute the particular production process steps within the sequence (Step 3). Then, the production processes are reordered to a sequence that seems optimal to fulfill the client's goals, such as the throughput (Step 4). Then, the engineers lay out the production resources in a shop floor design considering the requirements of the client, such as the available space (Step 5). Figure 1.6 shows a schematic section of a CPPS shop floor design with (a) robot cells in a work line with (b) buffers and storage for intermediate products that are connected via (c) transport facilities to other work lines, and (d) human workplaces in the CPPS. This step might already require re-planning the production process sequence of the constraints of the client cannot be fulfilled.

To this extent, our industry partner is aware of the *structural resource variability*. Our industry partner addresses this challenge by modularizing and qualifying technical components in their enterprise resource planning system. Furthermore, a representation of technical components and their parameters are stored in a spreadsheet, the so-called *technical library* with corresponding CAD pictures. Best-practice taxonomies for structuring resources along hierarchies also exist, which our industry partner aims to follow.[7] However, to the best of our knowledge, few tools support these taxonomies in practice.

The CPPS design with the production process steps and resource designs are later used to implement CPPS artifacts in the respective engineering disciplines, such as the control software. With this rough design, the engineers can further investigate the advantages and drawbacks of these CPPS design options, such as distribution over several CPPSs. Furthermore, they can create corresponding rough cost estimates for the realization of potential CPPS designs provided to the client. If customers accept a CPPS design and cost estimate, *basic engineering* hands over the artifacts to the different disciplines of *detail engineering*.

**Summary: Traditional Engineering.** The traditional engineering process is manually conducted by experienced engineers mainly based on their implicit knowledge. While this part of the engineering process might be straightforward to senior engineers in a single project, these issues *make the engineering process error-prone and hard to reproduce*. Missing holistic knowledge models that allow the modeling of the main concepts of CPPS engineering with their variability intensify this issue. Relying on *implicit knowledge significantly impedes knowledge transfer* to other colleagues and disciplines, making the reuse of concepts and artifacts difficult and inefficient. The lack of knowledge models for communicating engineering knowledge and systematic approaches for the reuse of engineering assets poses an essential problem. This *insufficient combination of multidisciplinary knowledge* further hinders supporting engineers with automating the engineering process.

Many engineering organizations have a strong focus on production resource planning and see production processes more as a means to an end to support resource selection. To this

[7]ECLASS Standard: `https://eclass.eu/eclass-standard`

extent, the PPR paradigm [181] provides a novel perspective into their CPPS engineering process with the benefit of connecting these concepts. However, the large variability of products, production processes, and production resources to choose from opens a vast configuration space with permutational complexity and *NP-hard* difficulty. This also poses an issue as the systems have long runtimes and maintenance spans, which require the propagation of reconfigurations or bug fixes. The lack of approaches to integrate these different dimensions of variability presents a significant problem.

Several engineering organizations that we collaborated with consider SPL engineering, including variability modeling, twofold. On the one hand, they welcome the expected opportunities that SPL methods and techniques promise. On the other hand, they are concerned about the additional complexity and effort that introducing SPL engineering might render in an already multidisciplinary engineering environment. Currently, the engineers typically *clone and own* [54] the engineering artifacts, for instance, spreadsheets with product types and their parts, including errors and semantic inconsistencies. Our industry partner aims to counter those issues with typical methods from classic software engineering, such as templates and modularization, adopted to CPPS engineering. These practices align with some of the current strategies and techniques used in the production systems industry and how SPL engineering is currently perceived [6, 9, 159].

### 1.1.3 Challenges

In the prior section, we mentioned that CPPS engineers encounter multifaceted challenges that impede their efforts toward more efficient CPPS family engineering. The traditional engineering approach is a predominantly manual process with limited artifact and hardly any tool support for variability management. This limitation impedes the seamless integration of knowledge and engineering expertise essential for designing and implementing efficient CPPS families. Furthermore, it leads to error-prone engineering artifacts typically carried over multiple projects, sometimes hampering cooperation with other disciplines and very limited reproducibility. This practice results in rework and high effort if project requirements change, which occurs regularly. In this context, the following challenges significantly impact reuse efficiency.

**CH1**. **Insufficient knowledge representation with variability in CPPS artifacts.** At the forefront is the challenge of *inadequate CPPS knowledge models*, related to **RL1**., that are easy to understand and manipulate for humans and, at the same time, machine-interpretable while supporting multiple disciplines. Furthermore, the systematic *representation of variability* within the engineering artifacts is deprived. Consequently, engineers struggle to express their *implicit discipline-specific knowledge acquired through experience* inherent in CPPS engineering that remains largely undocumented or is scattered in various heterogeneous artifacts. This implicit knowledge leads to opaque decision-making processes that are *hard and, most of the time, impossible to reproduce*, especially with their dependencies on engineering artifacts. Making this knowledge sufficiently explicit is still an open challenge in CPPS engineering [32, 33]. This issue also concerns the means to consistently model PPR concepts, for instance, using Domain-specific Languages (DSLs) or low-code approaches with sufficient semantics.

**CH2**. **Insufficient CPPS engineering applications for CPPS knowledge.** The shortcoming of insufficient knowledge representation largely influences the challenge of *insufficient knowledge management in CPPS engineering applications*, related to **RL2**., where knowledge is hard to constitute and reuse systematically. This challenge is exacerbated as the engineering process is often conducted in isolated information silos within the different disciplines. However, this inherent difficulty in capturing, formalizing, and composing implicit knowledge hampers the transferability and communication of expertise across disciplines, projects, and engineering seniority. Changes of the CPPS design, which occur frequently due to client requests [195], lead to redesigning the entire CPPS from scratch [130, 148] rather than employing *systematic and efficient reuse management.*

**CH3**. **Inefficient reuse and variability management in CPPS engineering.** While guidelines exist [200], practice shows [6, 159] that the concept of systematic reuse and variability management presents a significant challenge in CPPS engineering. As a result, engineers grapple with *weak understanding of evolving CPPS reuse and variability aspects.* For instance, current approaches in CPPS engineering often result in manual effort-intensive *clone-and-own* strategies [54]. Furthermore, the means to systematically configure assets based on reusable artifacts are unclear. This is especially true for assets that need to be revalidated after a reconfiguration due to, for example, product or batch changes. This *lack of systematic reuse mechanisms impedes efficiency* and limits innovation and agility in adapting to changing requirements toward the CPPS.
Nevertheless, CPPSs induce complex variability concerns that stem, in particular, from the three principal product families comprising (i) product types and their characteristics, (ii) production processes and their dependencies, and, (iii) production resources with their potential for reconfiguration. These families with the fourth dimension of CPPS families form a multi-product line [4, 78]. Together, this incorporates the variability of the PPR families, their dependencies, often non-functional constraints, and feature cardinalities. The variability spans a *vast complex design and configuration space* when planning the CPPS that demands sophisticated solutions for effective planning and execution [28, 155, 185]. However, an insufficient combination of multidisciplinary knowledge impedes the automation of the engineering process. Therefore, this variability must be managed efficiently to enable systematic reuse. A manual configuration is highly inefficient and tedious due to the large number of possible production sequences, which challenge engineers to find practically feasible ones [137]. Furthermore, modeling regarding production process sequence variability and characteristics is inaccurate due to missing dependencies to product and resource variability [117]. This weak understanding hampers designing CPPS process composition and optimization, leveraging production flexibility for a product family [117]. Meanwhile, current

research approaches often focus on *either* product variability [3, 152, 197] *or* process variability [176, 186], neglecting the interconnectedness between the two. Thus, this laborious, time-consuming, and error-prone activity calls for a methodology to document and automate production sequence and resource configuration derivation from a product configuration.

These three challenges delineate the scope of the thesis regarding efficient reuse and variability management for CPPS families.

Efficient CPPS Reuse and Variability Management.

Efficient CPPS reuse concerns establishing systematic approaches for multidisciplinary knowledge externalization and reuse. Additionally, it concerns systematic, semi-automated approaches for employing reusable engineering artifacts. These approaches aimed to overcome the largely manual, error-prone, and unreproducible traditional engineering process. Furthermore, they should help narrow down the CPPS configuration space and subsequently lower domain expert effort. Such approaches shall include managing the variability in engineering artifacts based on explicit CPPS engineering knowledge for PPR families.

Overall, these challenges hinder the *integration and seamless cooperation of disciplines* with iterative knowledge backflows from engineering and operation concerning multidisciplinary collaboration and knowledge exchange while *maintaining the necessary separation of concerns* between the disciplines. Overcoming these challenges requires an integrated approach that addresses the interplay between knowledge modeling capabilities, advanced engineering applications, reuse and variability management strategies, and multidisciplinary cooperation.

## 1.2 Related Work

This section discusses related work from the research fields CPPS and SPL engineering, presenting the research that this thesis builds on and the research gaps for this thesis.

### 1.2.1 CPPS Engineering

We mentioned that CPPSs are production systems to facilitate flexible, reconfigurable production of highly customizable products [81, 143].

Therefore, the Industrie 4.0 initiative has developed principles and architectures, such as the Reference Architectural Model industrie 4.0 (RAMI 4.0) [160]. The RAMI 4.0 provides a foundational framework to support the design, development, and integration of Industrie 4.0 solutions by combining three axes of engineering CPPS [139]. These axes are (i) the architecture axis –layers– defining the role of CPPS assets, (ii) the lifecycle axis representing the life cycle of CPPS assets, and (iii) the hierarchy axis for assigning

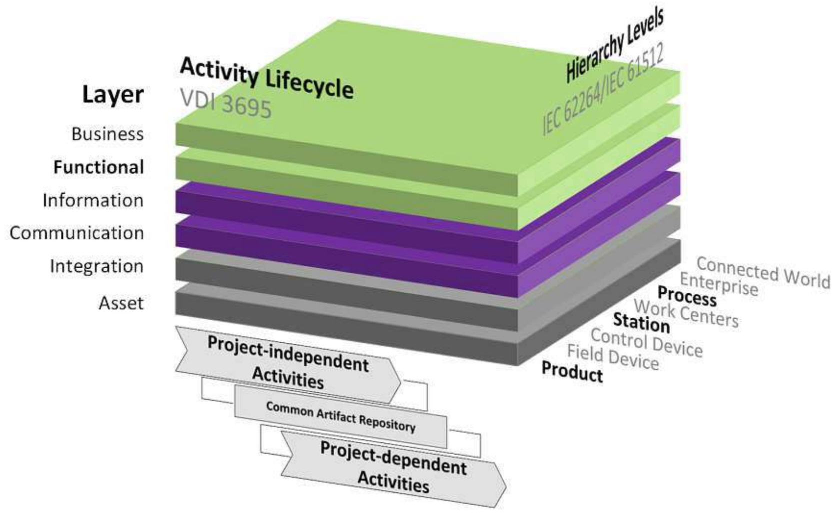


Figure 1.7: Context of the contributions of the thesis in the RAMI 4.0, based on RAMI 4.0 [160].

functional models of assets to individual levels [160]. Figure 1.7 shows an adapted version of the RAMI 4.0 setting the *scope of this thesis* (bold concepts) concerning (i) the *engineering* lifecycle phase, (ii) for *PPR concepts* in the hierarchical axis, (iii) on the *functional* layer of the RAMI 4.0.

Beyond the RAMI 4.0, the Verein Deutscher Ingenieure (Association of German Engineers) (VDI) provides several guidelines for structured plant engineering processes [198, 200] to tackle the complexity of production systems and their planning. The VDI 3695 guideline on evaluating and optimizing engineering industrial plants defines several measures. In particular, the VDI 3695 raises, beyond others, the following *measures* with specific target states for consideration.

M1. *Models and description languages* to create models of engineering concepts with syntax and semantics that adhere to the aspects described by a model, reaching from natural language and programming languages over graphical symbols and diagrams to low-code approaches, supporting the diverse requirements of disciplines concerning expressiveness, related to **RL1**..

M2. *Knowledge management* to systematically handle knowledge on information, findings, and experiences in formalized and machine-interpretable representation, related to **RL1**. and **RL2**..

M3. *Re-use* to facilitate the optimization of engineering (**RL1**.–**RL3**.).

M4. *Quality assurance* to ensure that products or services maintain specified levels of quality (**RL1**.–**RL3**.).

M5. *Integration and seamless cooperation of disciplines* and their activities across the engineering life cycle to ensure the coordination, consistency, and interoperability of results while separating their individual concerns related to all three research lines (**RL1**.–**RL3**.).

*This thesis considers these VDI 3695 measures as opportunities for research* aiming to provide significant contributions to each of these measures.

In CPPS engineering, the engineers within and over the disciplines work on and share various engineering artifacts. They reuse several artifacts to improve engineering quality and efficiency and lower the engineering effort and risk [71, 178]. Artifacts like Type Comparison Matrices (TCMs) already contain information on product variants and implicit production processes [44]. Nevertheless, these TCMs induce further issues, such as the lack of semantics and standardization. However, *we utilize TCMs as an example regarding the relevant variability of products* in industrial engineering artifacts [43].

Concepts and artifacts that focus on functional CPPS models are, for instance, the PPR approach [181, 199] and Capability- and Skill-based Engineering (CSBE) [58, 93, 149], providing foundational models for CPPS engineering knowledge.

The PPR approach [181], denoting products, production processes, and production resources as first-class-citizen concepts, is a well-established concept for production systems planning [58]. The VDI 3682 guideline [199] for process descriptions, named Formalised Process Description (FPD), formalizes this approach. While Part 1 of the guideline provides a concise and straightforward visual notation, Part 2 describes a technology-agnostic knowledge model. This way, the VDI 3682 allows modeling products, production process sequences with input and output products to manufacture the products, and production resources that execute the processes. Figure 1.5 shows a section of such a VDI 3682 model that visually describes the *insertion of the pipe and a lock* (circles) into a production system and the process of *welding the lock onto the pipe* (rounded rectangle). The model also shows the production resources (rounded rectangle) that execute those processes, such as the *linefeed* and the *laser welding robot.* Other knowledge models, such as AutomationML (AML) [32] or the Planning Domain Definition Language (PDDL) [61], are able to represent production processes. The first approach concentrates more on data exchange, while the second one focuses more on scheduling single production processes and does not support reconfiguration and adaptation easily. However, they do not provide a visual presentation, which we deem an important characteristic for the acceptance of the domain engineers. Therefore, *we build on the VDI 3682 to model the PPR aspects* of CPPSs with a focus on production processes as first-class citizens.

Part 3 of the VDI 3682 guideline aimed at an Extensible Markup Language (XML) representation of the knowledge model. However, even though the VDI first published the guideline in English in 2005, is still in preparation and not expected before the third quarter of 2025, posing a significant limitation.[8] Furthermore, it does not provide the means to express the needed variability of the concepts. Therefore, *we exploited the gap*

[8] VDI 3682 Part 3: `https://www.vdi.de/richtlinien/details/vdivde-3682-blatt-3-formalisierte-prozessbeschreibungen-xml-repraesentation`

*of a missing human-interpretable and machine-readable VDI 3682 format* to develop a concise DSL that supports the modeling of variability for PPR concepts and their dependencies. For the development of the DSL, we initially used the guideline of Strembeck and Zdun [191] specifying a concrete and abstract syntax and providing a corresponding meta-model according to the Meta Object Facility (MOF).[9] In subsequent work and additional publications, we followed more advanced language modeling frameworks (cf. Section 3.1.2).

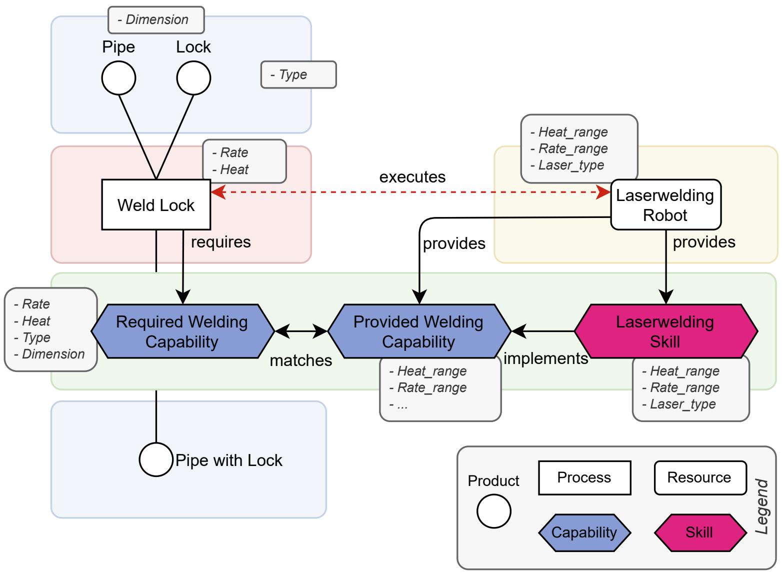


Figure 1.8: PPR model for a *welding process* in VDI 3682 [199] notation *without* (dashed connection *executes*) and *with* Capabilities and Skills (C&Ss), based on Meixner et al. [135].

Fostering reuse requires decoupling of the production process from resource modeling regarding the VDI 3682 and proposed DSL. CSBE [58, 93, 149] uses C&Ss. It aims at decoupling the *required capabilities* for production processes from the *provided capabilities* of production resources. Further decoupling is achieved by instantiating a provided capability as a *skill implementation* for a particular resource type. For instance, Figure 1.8 shows a PPR model for a welding process without (dashed connection) and with C&Ss (depicted as diamonds). In this example, the *Pipe* and the *Lock* need to be welded together, so *Welding* is a capability that is required by a production process with a certain speed *rate* and *heat*. This required capability can be matched to a group of provided capabilities that support the ranges of the speed *rate* and *heat*. Going further, the provided capability can be specialized to a *Laser Welding* capability, which is implemented for a particular robot arm in the control code. *We build on the CSBE concept* to describe applications of CPPS knowledge management and model decoupled reusable PPR assets.

[9]Meta Object Facility: `https://www.omg.org/mof`

### 1.2.2 SPL Engineering

SPL engineering is founded on mass customization of (software) products utilizing a shared platform of reusable artifacts with individually configurable features [25, 152]. Pohl et al. [152] introduced the *DAE* framework (cf. also Figure 1.10) based on a framework for software construction with components [144, 203] with two life cycles. The *domain engineering cycle* focuses on developing “artifacts for reuse”, shared in a common artifact repository [197]. The *application engineering cycle* concentrates on the development “with reusable artifacts” that are instantiated and configured for particular requirements. *This thesis exploits research opportunities within the DAE framework* regarding its application for CPPS engineering.

Within SPL engineering, *variability modeling* is a crucial activity to capture the commonalities and differences of the investigated system. Therefore, the research community has developed various variability modeling methods and models investigated in a plethora of Systematic Literature Reviews (SLRs), mapping studies, and surveys [5, 8, 24, 29, 156]. The de-facto standard for variability modeling are feature models [86]. Other important variability modeling approaches are decision models [30, 68], and orthogonal variability models [152]. These approaches have been refined and extended in the last decades [29, 156]. More recently, the Universal Variability Language (UVL) [193], a textual DSL for variability modeling, was developed as a community effort and gained attention.

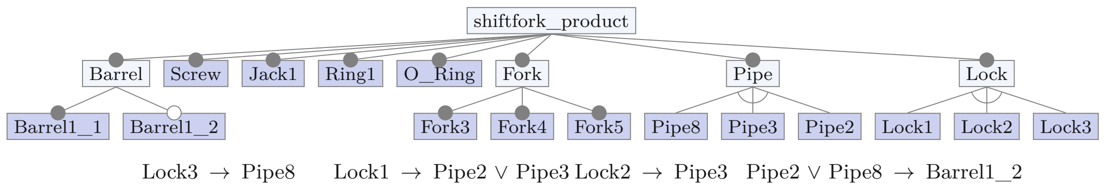


Figure 1.9: Feature model of a *shift fork* (cf. Section 1.1.1) [137] (cf. also Figure 3.13a).

**Feature models** Feature models have rooted in the Feature-Oriented Domain Analysis (FODA) method and elicit commonalities and variable features in a feature tree with cross-tree constraints, i.e., inclusion and exclusion criteria [86]. Figure 1.9 shows the feature model of the shift fork product family of the *shift fork* use case (cf. Section 1.1.1), conveying the same information as the TCM (cf. Table 1.1) but semantically unambiguous. The model consists of *mandatory* features representing product parts required in all variants, such as a *Screw*. The model also contains *optional* features, such as *Barrel1_2*. Furthermore, it contains feature groups, such as the alternative *Pipe* group that allows selecting only one type of pipe. While feature models are well suited to represent structural variability, since hierarchy is a key concept in these approaches, they are not designed to define behavioral variability, i.e., configuration sequences. This also means that the user guidance during their configuration is relatively low [138], which has been shown helpful in product configuration [158]. *This thesis builds on feature models* to represent the structural variability of CPPS aspects. At the same time, *we exploit the inability of feature models to represent configuration sequences* in this thesis.

| ID | Question | Type | Range | Card. | Visible/Relevant if | Constraint/Rule |
|---|---|---|---|---|---|---|
| Pipe | Which Pipe type? | Enum | Pipe2 \| Pipe 3 \| Pipe8 | 1:1 | false | |
| Lock | Which Lock type? | Enum | Lock1 \| Lock2 \| Lock3 | 1:1 | false | Lock1 $\implies$ Pipe = Pipe2 $\vee$ Pipe = Pipe3<br>Lock2 $\implies$ Pipe = Pipe3<br>Lock3 $\implies$ Pipe = Pipe 8 |
| Barrel1_2 | With Barrel? | Boolean | true \| false | | Pipe2 $\vee$ Pipe8 | |

Table 1.2: DOPLER Decision Models (DM) [30] representing the *shift fork* product variability.

**Decision models** Decision models are rooted in the Synthesis method and elicit only variable decisions in a decision table with their constraints [23]. For instance, the DOPLER approach [30] comprises decision models and asset models for defining variability. With their visibility conditions, they are also capable of describing configuration sequences. Table 1.2 represents an exemplary DOPLER decision model in tabular representation. A decision in a decision model consists of a unique `ID` and a text describing the decision (usually a `Question`). These decisions are configured based on the specified `Range`. For instance, only one of the three locks can be selected in the enumeration decision *Lock*. `Constraint/Rule` and `Visible/Relevant if` relationships between decisions specify *(post-)conditions* and hierarchical or logical *(pre-)conditions* defining orders of making decisions during the configuration. The `Visible/Relevant if` relationship defines preconditions that need to be satisfied for the decision to be selectable. For instance, *Barrel1_2* can only be selected if *Pipe2* or *Pipe8* is selected. These explicit dependencies among selected decisions reflect a configuration order, which models behavioral variability. *This thesis builds on decision models* to represent the behavioral variability of CPPS aspects.

### 1.2.3 CPPS and SPL Engineering

The VDI 3695 also introduces a procedure model (cf. Figure 1.10) that defines Project-independent Activities (PIAs) and Project-Dependent Activities (PDAs) activities that structure and formalize the engineering process [200]. The VDI 3695 procedure model originates from the DAE framework. To briefly recap, the PIAs start with a domain analysis, continue with the planning and realization (engineering) of reusable domain artifacts, and conclude with reusable, configurable, tested, and approved artifacts. The PDAs start with the acquisition and requirements elicitation of a CPPS project and continue with the planning of the CPPS employing and configuring the reusable artifacts, CPPS realization, and finish with the ramp-up and commissioning of the CPPS for handover to operations.

Jazdi et al. [83] described the combination of these procedure models and details on the PIAs for production systems engineering. In their work, Jazdi et al. [83] call for

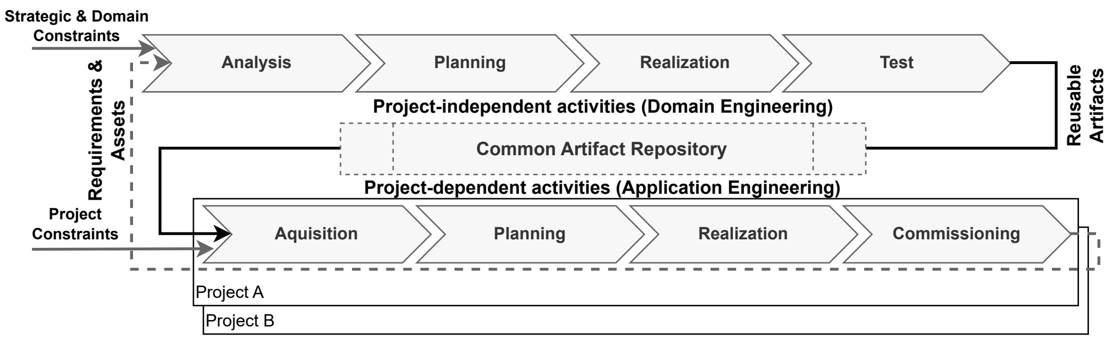

Strategic & Domain Constraints
Analysis
Planning
Realization
Test
Requirements & Assets
Reusable Artifacts
Project-independent activities (Domain Engineering)
Common Artifact Repository
Project-dependent activities (Application Engineering)
Project Constraints
Aquisition
Planning
Realization
Commissioning
Project A
Project B

Figure 1.10: Domain and Application Engineering (DAE) cycle in SPL and CPPS engineering [83, 152, 200].

action to adopt and implement research achievements. This adoption includes the SPL engineering and CPPS engineering domain to gain, beyond others, from the following SPL *capabilities*.

C1. *Efficient Reuse* of assets across various CPPS configurations, leading to significant efficiency gains in development and maintenance, related to **RL1**. and **RL3**..
C2. *Adaptability* accommodating changes in production requirements, technology advancements, and market demands more effectively, related to **RL3**..
C3. *Variability Management* managing variability in software-intensive systems to handle the diverse configuration options and production scenarios related to **RL1**. and **RL3**..
C4. *Enhanced Quality and Consistency* of components reused across CPPS instances, mitigating the risks of errors and inconsistencies (**RL1**.–**RL3**.).
C5. *Facilitation of Multidisciplinary Collaboration*[10] encouraging the collaboration between engineers from diverse disciplines (**RL1**.–**RL3**.).

*This thesis considers these SPL capabilities for CPPS engineering as opportunities for research* aiming to provide significant contributions to each of these capabilities adopted to CPPS engineering.

In the introduction, we reported on the different dimensions of variability in CPPSs regarding families of products, processes, and resources that together build a multi-product line [55, 147, 196]. Within the PPR modeling concept, production processes play a central role [199], similar to production system engineering in general [55, 147]. This type of variability requires modeling preconditions for process steps that need to be executed subsequently [59].

[10]The original publications (cf. [83, 152, 197] often use the term "interdisciplinary collaboration" when referring to the benefits of SPL engineering. To fit the wording of this thesis, we use the term "multidisciplinary collaboration".

Regarding production processes and SPLs approaches, several SLRs report that variability of process models, such as business processes, are underrepresented in SPL research [156, 59]. Rombach [174] raised integrating software processes and product lines as future work. Feature models, as single models, are incapable of representing the essential behavioral variability. However, DOPLER decision models [68, 30] facilitate such constructs via visibility conditions *that we employ to represent the process preconditions.* Nevertheless, existing variability modeling approaches and models focus either on structural *OR* behavioral variability. As a result, current approaches are *as-is* incapable of handling the combination of different CPPS product families. Consequently, *we address the research gap of missing process configuration support* for integrated structural *AND* behavioral variability.

The dimensions of CPPS variability entail, among other things, the modularization of variability models and the separation of concerns between the stakeholders that model the different families but also the ones that configure them. Holl et al. [78] reported in an SLR on a plethora of works that address multi-product lines [162, 175] and multi-view product lines [80, 145]. Staged configuration [26, 108] or multiple levels of variability models [27, 162], so the subsequent configuration of multiple self-contained but interdependent variability models is one way to tackle multi-product lines. These approaches can be combined, for instance, with configuration flows [79, 141, 142] to coordinate the particular sequence of the variability model configuration. This requires establishing cross-model constraints to the express dependencies between the variability models [20]. In this thesis, we pick up the concepts of multi-product and multi-view families. Therefore, Linsbauer et al. [106] allows propagating feature dependencies automatically among several feature models. However, to the best of our knowledge, these works use feature models and do not allow the modeling and configuration of process sequences within a single variability model. Therefore, *this thesis aims to address the research gap of multi-product and multi-view families with configuration sequences* within one variability model.

For Cyber-Physical System (CPS) and CPPS in particular, several works support multi-product and multi-view families. Safdar et al. [179] developed a framework for the e CPSs domain that uses Unified Modeling Language (UML) and Object Constraint Language (OCL) constraints for variability modeling. Similar to the approaches mentioned in the previous paragraph, the approach *does not support process sequence configuration presenting a research opportunity.*

SysMl v2 is a systems modeling language, currently undergoing finalization, that uses a visual and textual representation, with its predecessor being a well-established standard[37]. Epp et al. [37] present an approach to use SysML v2 as a modeling language for expressing the variability of production systems. To this end, they incorporated the concept of feature models into SysML v2. However, *they do not consider behavioral variability* as required for production processes.

In her thesis, Fang [40] developed a multi-product and multi-view approach for the production systems domain, including the topology of a system, which requires process variability. The approach combines a topology and a process to define the dependencies between the different views. However, they rely on templates for the process models without the possibility to define the degrees of freedom outside these templates. This means that only between templates that represent a certain process variability can be chosen. In contrast, *we aim at addressing the degrees of freedom of the production processes during the configuration*.

Fadhlillah et al. [39] developed a multi-product and multi-view variability management approach for CPPSs. In this approach, engineers can also choose which type of variability model they want to use for each single product line. However, they *concentrate more on the Business-Architecture-Process-Organization (BAPO) layers* of CPPS variability, *considering processes as the configuration workflow* for the multi-product configuration, rather than the PPR approach and production process variability within a single model. Nevertheless, throughout this thesis, the author of the work and the author of this thesis closely collaborated.

**Summary: Related Work**

The related work highlights the foundational frameworks, methodologies, and techniques in CPPS and SPL engineering, emphasizing their strengths and limitations. These frameworks already provide essential structures for flexible and efficient production. However, significant gaps remain, particularly in externalizing engineering knowledge and propagating it to the different involved engineering disciplines. Furthermore, gaps remain for modeling the variability of the combined PPR concepts and an integrated approach for managing knowledge for reuse and variability. This thesis aims to address these gaps by developing novel approaches and mechanisms that enhance the flexibility, reconfigurability, and reusability of CPPSs, advancing state-of-the-art in both research domains.

## 1.3 Research Goals

By tackling these impediments head-on, CPPS engineers can pave the way for more efficient, reconfigurable, and sustainable CPPSs that meet the demands of today's dynamic manufacturing landscape. This mainly concerns automating the derivation of production sequences and resource configurations from a product configuration.

From the research lines (**RL1**.–**RL3**.), we derive the following research goals (Gx).

**G1**. **PKMO.** Design knowledge models aligned with state-of-the-art approaches that allow modeling various aspects of CPPSs. These models shall provide the foundation for explicitly capturing and representing CPPSs knowledge and its reuse in CPPS engineering and operation (**RL1**.). Furthermore, they shall provide the basis for the reuse of multidisciplinary knowledge for applications, such as advanced data analytics and production quality improvement.

**G2**. **ACEA.** Develop models and methods for CPPS engineering applications. These approaches shall support engineers in iteratively conducting DAE activities based on the previously designed knowledge models to improve engineering knowledge coordination and communication (**RL2**.). Furthermore, models and methods models shall improve the multidisciplinary collaboration throughout the engineering process.

**G3**. **IRVM.** Establish explicit and systematic knowledge modeling and engineering applications for selected DAE activities for the PPR concepts utilizing techniques and technologies from knowledge modeling and management, CPPS engineering, and SPL engineering for integrated reuse and variability management to facilitate adaptability and flexibility of CPPSs (**RL1**.-**RL3**.).

**G4**. **Prototypical Applications.** Implement prototypical applications, as proof of concept, to validate the knowledge models and engineering applications developed for reuse and variability management within CPPS engineering (**RL1**.-**RL3**.).

**G5**. **Empirical Evaluation.** Showcase successful knowledge application through empirical evaluation backed by real-world use cases as a demonstration of the practical applicability and effectiveness of the approaches providing evidence to support our assumption of more efficient reuse and variability management for CPPS engineering (**RL1**.-**RL3**.).

## 1.4 Research Methodology

The research methodology of this thesis follows the iterative *Design Science* approach [73, 74, 205]. The approach embeds *information systems research*, and in more general *software engineering research*, into a broader framework that includes its related *environment* and its foundational *knowledge base*.

Figure 1.11 shows an instantiation of the Design Science methodology for this thesis. The design science methodology defines three cycles for conducting research on information systems: the *relevance* cycle, the central *design* cycle, and the *rigor* cycle [73, 74]. Its primary goal is to *design artifacts that treat a problem in the particular context.*

The following section describes the technological rule as an aid to guide the application of the design science methodology. Furthermore, it describes the Design Science cycles' activities conducted in the context of this thesis.

### 1.4.1 Design Science Technological Rule

In information systems and software engineering research, a technological rule captures the essence of design science [36]. It does so by defining issues based on what stakeholders demand through interventions in a particular context [36]. Such a generalization shall help to identify and communicate a research project's practical impacts [36]. Therefore, we formulate the following technological rules to guide the application of the design

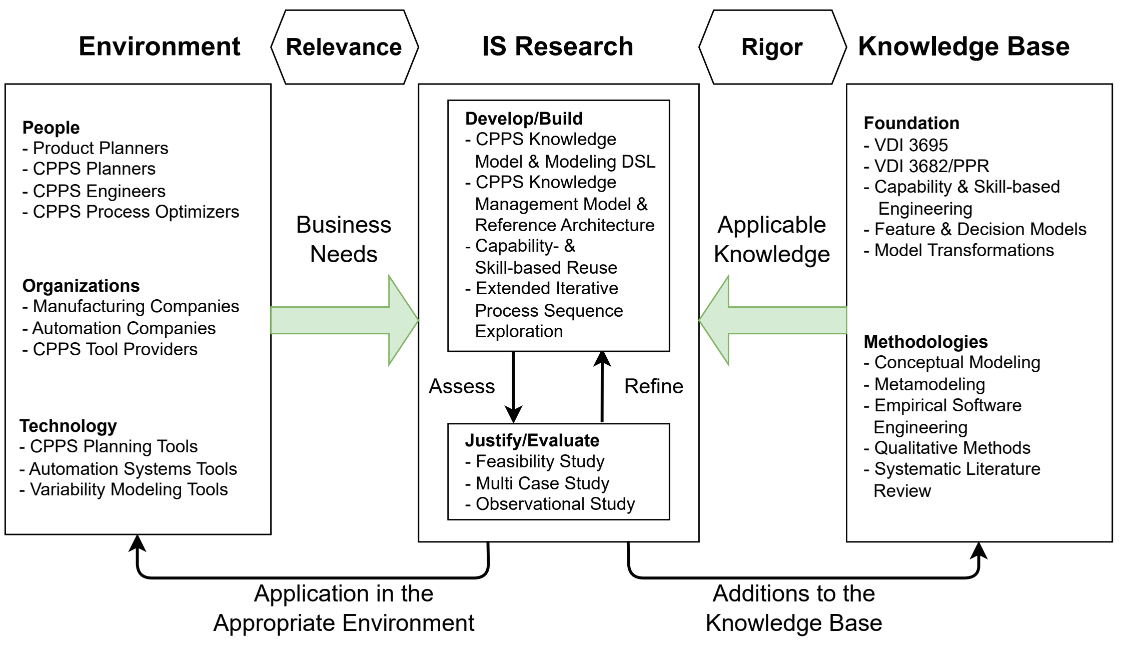

Environment
Relevance
IS Research
Rigor
Knowledge Base
People
- Product Planners
- CPPS Planners
- CPPS Engineers
- CPPS Process Optimizers
Organizations
- Manufacturing Companies
- Automation Companies
- CPPS Tool Providers
Technology
- CPPS Planning Tools
- Automation Systems Tools
- Variability Modeling Tools
Business Needs
Develop/Build
- CPPS Knowledge Model & Modeling DSL
- CPPS Knowledge Management Model & Reference Architecture
- Capability- & Skill-based Reuse
- Extended Iterative Process Sequence Exploration
Assess
Refine
Justify/Evaluate
- Feasibility Study
- Multi Case Study
- Observational Study
Applicable Knowledge
Foundation
- VDI 3695
- VDI 3682/PPR
- Capability & Skill-based Engineering
- Feature & Decision Models
- Model Transformations
Methodologies
- Conceptual Modeling
- Metamodeling
- Empirical Software Engineering
- Qualitative Methods
- Systematic Literature Review
Application in the Appropriate Environment
Additions to the Knowledge Base

Figure 1.11: The applied Design Science approach in the context of this thesis [73, 74, 205].

science methodology for this thesis and relate them to the research lines (**RL1**.–**RL3**.) the VDI 3695 *measures* (M1.–M5.) and SPL *capabilities* (C1.–C5.).

*Design Science Technological Rule 1.* To achieve **(situation)** in *multidisciplinary CPPS engineering* (M5., C5.) **(effect)** *explicit production knowledge modeling* (M1., M2.) of CPPS assets with enhanced quality and consistency (M4., C4.) and *improved CPPS knowledge coordination and communication* (M2., M5., C5.), **(intervention)** apply the *PPR knowledge model*, the *Product-Process-Resource Domain-Specific Language (PPR–DSL)*, and *advanced CPPS engineering applications* (**RL1**., **RL2**.).

For instance, engineers shall be able to express their knowledge of a particular welding process or a welding robot in a human-readable and machine-interpretable model when planning a welding work line for car parts. Therefore, they shall be able to follow a systematic approach to elicit existing knowledge from engineering artifacts of prior projects. Additionally, the engineers shall be able to suitably communicate the knowledge of their models to stakeholders of multiple other disciplines. The engineers shall be able to conduct this task, for instance, with an engineering design pattern.

*Design Science Technological Rule 2.* To achieve **(situation)** in *multidisciplinary CPPS engineering* (M5., C5.) **(effect)** *efficient reuse and variability management* (M3., C1., C3.) of CPPS assets with enhanced quality and consistency (M4., C4.), **(intervention)** apply the *integrated reuse and variability management* approach (**RL3**.) employing the PPR knowledge model and advanced CPPS engineering applications.

The engineers shall be able to reuse the previously defined models and artifacts efficiently, for instance, to reconfigure the work line if new products are required. For example, engineers shall be able to model the new product, such as an additional shift fork with a particular welding seam length. Then, they shall be able to reuse an already defined welding process or robot template and easily "inject" the welding seam length of the shift fork to get a deployable engineering artifact, such as robot control code.

From these rules, we can derive the particular methodological components for this thesis.

### 1.4.2 Relevance Cycle

The *relevance cycle* links the research project's *contextual environment* to the activities of the design cycle [73, 74]. It investigates the *business needs* and elicits *requirements* for research as well as *evaluation criteria* for the design artifacts [73, 74]. The research project's results shall be applied in the context.

Flexible production with reusable assets in CPPS engineering is an ongoing interdisciplinary topic in research and the industry [10, 40, 58, 84, 104]. The thesis is based on investigating CPPS engineering in the context of the long-running research project CDL SQI with several industry partners. Identified stakeholders relevant to this thesis on an organizational level are manufacturing or factory automation companies and CPPS tool providers.

We elicited challenges, requirements, and evaluation criteria from several industry partners and their use cases. These activities were conducted in continuous meetings and multiple workshops over several years that *the author of this thesis endeavored and actively led.* These industry partners have designed and engineered CPPSs or have aimed to provide information and engineering systems to individual stakeholders, such as product and CPPS planners, CPPS engineers, and production process optimizers. They require techniques and tools to tackle the inherent issue of implicit knowledge, knowledge reuse, and variability management with subsequent configuration. Existing techniques and technologies are tools and information systems for CPPS engineering, variant modeling, component libraries, and internal best practices for artifact reuse. However, these techniques and technologies often have limited capabilities that must be extended.

Furthermore, we verified, discussed, and investigated the identified challenges, requirements, and evaluation criteria with partners from research collaborations (cf. Feichtinger et al. [49, 50]). These collaborations partially existed in the research group or were *proactively established by the author* throughout this thesis in several meetings, workshops, or conferences. Additionally, to further gather an overview of research publications, *the author of this thesis initiated and conducted* a concise literature survey with authors from several research initiatives [58].

### 1.4.3 Rigor Cycle

The *rigor cycle* links the knowledge base of the research project's scientific fields to the activities of the design cycle [73, 74]. The knowledge base provides the theoretical *foundations* and research *methodologies* to conduct the research project rigorously [73, 74]. The research project's results add to the knowledge base via dissemination, including the publication of research data, presentations, workshops, conference sessions, papers, and this thesis.

This research project aims to support CPPS engineering through efficient reuse and variability management with explicit knowledge management. To this end, the relevant research fields concern *CPPS Engineering*, *SPL Engineering*, and *MBSE* integrating their foundations and methodologies. Hence, the thesis aims to utilize methods and techniques established in these communities. In particular, it utilizes frameworks and methods from CPPS engineering, such as VDI 3695, VDI 3682, and CSBE. Furthermore, it employs state-of-the-art feature and decision models from SPL engineering, as proposed in, for instance, Galster et al. [59], Raatikainen et al. [156], and model-based artifact transformations and agile methods from software engineering. To evaluate the design artifacts, we aimed to apply methodologies from empirical software engineering [52, 214], such as systematic literature reviews, conceptual modeling, and qualitative and mixed methods.

We worked with external and international researchers and domain experts to achieve distinguished research results and validate the methods and techniques. These activities aimed at exchanging our ideas and gathering feedback, for instance, on scientific conferences. There, *the author of this thesis acted as a special session and workshop organizer* in a leading role to foster research dissemination. Furthermore, we collaborated with research organizations that investigate complementary challenges and can provide orthogonal approaches where *the author took over the conceptual planning and overview*.

### 1.4.4 Design Cycle

Two pivotal research activities in the *design* cycle are conducted iteratively. The *artifact design* to improve a problem in a particular context, and the *artifact evaluation* to evaluate and validate the artifacts in this context [73, 74]. Together, they aim to provide short feedback loops to stakeholders from the environment to further refine the design [73, 74].

#### Artifact Design

The *design* activity creates purposeful artifacts that aim to solve significant problems in the environment [73, 74]. These artifacts are not fully mature systems ready for use but represent new ideas in concepts, models, methods, and techniques [73, 74]. They shall support the analysis, design, implementation, and utilization of information systems [73, 74].

We designed a set of artifacts comprising concepts for explicit knowledge management that support reuse and variability management methods and techniques. Furthermore, we designed and developed prototypical applications as a proof of concept for the approaches. Some of the most important artifacts are listed below following the *research lines* (**RL1**.–**RL3**.) and related to the VDI 3695 *measures* (M2.–M5.) and SPL *capabilities* (C1.–C5.).

**A1.1**. A *CPPS engineering knowledge model*, that the author developed and implemented, based on the VDI 3682 guideline supporting reuse and variability management. With such a knowledge model, we provide an engineering knowledge representation of the PPR approach [181] and establish the basis for reusable CPPS assets (M1., M3., M5., C5.).

**A1.2**. A *human-understandable and machine-interpretable DSL for the CPPS knowledge model*, where the author led the design and implementation process. Such a DSL provides the measures to explicitly define CPPS engineering knowledge with variability of domain experts from different disciplines over the CPPS lifecycle (M1., M3., M5., C5., C3.).

**A2.1**. A *model for advanced CPPS engineering applications* on CPPS assets with their dependencies to other assets, their characteristics, and data that the author developed (M2., M1., M3., M5., C5.). Such a model provides the foundation for iterative and traceable modeling in an engineering graph by identifying assets suitable for reuse.

**A2.2**. An *integration of C&S-based concepts* [93, 149] into the *knowledge model* for improved CPPS asset reusability that the author pursued (M2., C1.). The integration shall facilitate the loose coupling of processes and resources, making it easier to redesign in the engineering phase and reschedule during operation (M1., M3., M5., C5.).

**A2.3**. A *CPPS reference model* for the domain and its asset types to facilitate the *identification of engineering design patterns* for the reuse of components from assets (M2., M1., M3., M5.) that the author designed and established. Such a reference model shall guide CPPS engineers in finding and delimiting a cumulation of assets that can be reused as building blocks during engineering and operation, also serving improved coordination and communication. The development comprises the means to derive an engineering graph from the reference model as a foundation for advanced method integration, such as quality assurance or process optimization.

**A2.4**. A *systematic method for C&S reuse activities* over the CPPS engineering lifecycle aligned with DAE. The method is supported by loosely coupling PPR, in particular, production processes and resources with C&S (M2., M1., M5.) that the author created. This method shall guide engineers in how to elicit, organize, reuse, and configure reusable PPR assets from CPPS artifacts loose coupling with C&S a foundation for more efficient knowledge reuse.

**A3.1**. *Mapping rules and model transformations* from the CPPS model to state-of-the-art variability models (M1., M3., C1., C3.), where the author provided the domain-specific foundation. Such rules and transformations build the foundation to support engineers in their discipline while modeling and validating the variability in well-established models.

**A3.2**. *Documented and modeled industrial use cases* from the CPPS domain, including their variability, as a foundation for empirical evaluations of the different approaches that the author elicited, modeled, and documented. Furthermore, the use cases shall act as a possible baseline for the research community in the mind of open science and research reproducibility.

**A3.3**. An *iterative integrated approach* to reuse, configure, and validate CPPS assets and their dependencies, in particular, to configure products, explore the sequence of required production processes as a source of significant complexity, and configure production resources in CPPSs engineering (M2., M1., M3., M5., C1., C3., C5.) that the author designed and developed cooperating with a research partner.

**A3.4**. A *toolchain for integrated reuse and variability management* method and generate engineering artifacts from configurations while maintaining the separation of concerns between CPPS engineers and the engineering lifecycles (M2., M1., M3., M5., C1., C3., C5.), where the author supervised the conceptual design and led the development and implementation.

#### Artifact Evaluation

In the evaluation activity, the designed artifacts shall be evaluated, for example, for functionality, quality, and efficacy. The methodologies employed shall be founded in the scientific knowledge base and range from conceptual to more sophisticated evaluations [73, 74]. To this end, the evaluation of the artifacts shall provide empirical evidence to support the hypothesis raised for this thesis.

We utilized the following evaluation methodologies for the design artifacts in this activity to showcase successful knowledge applications.

- *Rigorous domain analyses.* with stakeholders from the CPPS domain for insights into the particular requirements and challenges within the field. This analysis should ensure a thorough understanding of the context based on industrial use cases [75, 187].
- *Engineering and internal validation.* Iterative engineering of the design artifacts with internal validations in the research team and researchers from collaborations.
- *Empirical studies.* Utilization of empirical studies, such as *Model and quality checks* utilizing and investigating the developed artifacts to provide empirical evidence of their efficacy and utility [52, 214]; *Case studies.* Application of the developed artifacts in multiple real-world case studies to assess their efficacy and utility across diverse scenarios, validating their applicability and robustness [177, 214]; and *A user study* in a lab setting with domain experts to investigate how they use the artifacts and receive feedback for iterative improvements and enhancements [52, 184].

### 1.4.5 Research Questions

The thesis explores how to improve CPPS engineering through models, methods, and techniques founded in the relevant scientific fields. In particular, it investigates: (1) Explicit knowledge modeling through a PPR knowledge model with a corresponding DSL.

(2) Advanced CPPS engineering applications for coordination and communication. These applications shall utilize a knowledge model for connected engineering assets and a reference model for identifying engineering design patterns. (3) An integrated approach for reuse and variability management to manage the variability of reusable artifacts of CPPS engineering as the foundation for efficient reuse. To this end, we derive the following research questions.

**RQ1**. *Production knowledge model for CPPS engineering.* What production knowledge model effectively captures the fundamental PPR aspects and their inherent variability and what techniques help express these aspects to facilitate knowledge externalization by engineers in multiple interacting domains?

**RQ2**. *Knowledge management for advanced CPPS engineering applications.* What knowledge management model, CPPS reference model, and design patterns for advanced CPPS engineering applications effectively organize and integrate CPPS assets and artifacts, their dependencies, and data to facilitate traceable engineering and knowledge coordination, serving as a foundation for systematic and effective asset reuse?

**RQ3**. *CPPS family reuse and variability management.* What is an effective and efficient approach for reuse and variability management that accurately represents *the structural and behavioral variability* inherent in assets of CPPS families, and how can such an approach guide engineers in modeling reusable artifacts to derive valid CPPS configurations and artifacts?

### 1.4.6 Research Contributions

This section presents the main contributions of the thesis, addressing the identified challenges in CPPS engineering (**CH1**.–**CH3**.). It highlights advancements in knowledge modeling, engineering applications, and integrated reuse and variability management approaches, aiming to enhance effectiveness and efficiency in CPPS design. To this end, we seek to empower CPPS engineers with the means to facilitate knowledge externalization and communication across disciplines. These means should provide the foundation for reusable CPPS artifacts with their variability, fostering innovation and agility. The main contributions of this thesis, in brief, are:

**CO1**. *A production knowledge model for CPPS engineering and operation:* Development of a comprehensive production knowledge model for CPPS engineering and operation, capturing the essential aspects of PPR extended for reuse and variability management. *A DSL for CPPS engineering and operation:* Development of a human-readable and machine-interpretable DSL based on the knowledge model that enables engineers to define CPPS engineering knowledge explicitly. This contribution relates to **RQ1**. and comprises design artifacts **A1.1**. and **A1.2**..

**CO2**. *A knowledge model for advanced CPPS engineering applications:* Development of an extended production knowledge model as a foundation for coordination and communication of multidisciplinary engineering knowledge. *A knowledge management approach for advanced CPPS engineering applications:* Development of a systematic approach for CPPS asset reuse providing a reference model for CPPS engineers with a blueprint for engineering design patterns facilitating communication and collaboration across multiple disciplines. *A systematic method for C&S reuse:* Development of a method for CPPS engineering structured as DAE activities for eliciting, developing, and configure reusable PPR assets with C&S for loose coupling fostering efficient knowledge reuse This contribution relates to **RQ2**. and comprises design artifacts **A2.1**. to **A2.4**..

**CO3**. *An integrated reuse and variability management approach:* Design and implement an iterative, integrated approach for variability management of assets for CPPS families. This approach shall enable engineers to design PPR model and configure products, explore production process sequences, and configure production resources efficiently and validated as a foundation for artifact generation. This development entails the creation of mappings for an integration of the CPPS knowledge model with state-of-the-art variability models, such as feature models and decision models. This contribution relates to **RQ3**. and comprises design artifacts **A3.1**. to **A3.4**..

## 1.5 Outline

Chapter 1 introduced the context of the thesis regarding engineering CPPS to manufacture product families flexibly and the necessity of systematic reuse and variability management. Furthermore, the chapter presented the *shift fork* and *rocker switch* use cases for illustrative purposes to better explain the traditional approach of CPPS engineering. Then, the chapter introduced the research goals and the employed research methodology, including the research questions and design artifacts.

Chapter 2 gives a synopsis of the ten core publications contributing to this cumulative thesis grouped by the three research lines (RL.1) Production Knowledge Models, (RL.2) Advanced CPPS Engineering Applications, and (RL.3) IRVM for CPPS engineering. To this end, the publications are first presented chronologically in a table showing the corresponding venue, the author's position, and the research questions the particular publication aims to answer. Then, the chapter summarizes each publication and its contributions.

Chapter 3 discusses the scientific contributions of the thesis author, grouped by the research lines. The chapter, first, presents production knowledge models that structure the CPPS domain knowledge for representing PPR aspects with variability. Subsequently, it presents the PPR–DSL for externalizing CPPS engineering knowledge. Second, the chapter introduces advanced CPPS engineering applications. These provide means to structure the externalized knowledge with a reference model, communicate it via engineering design patterns, and systematically reuse engineering artifacts along the DAE activities with PPR and C&S. Third, the chapter discusses integrated reuse and variability management for CPPS engineering as a semi-automated, systematic, and integrated approach to managing the variability of reusable artifacts of CPPS families as a foundation for efficient reuse.

Chapter 4 discusses and concludes the thesis regarding the research results and outcome. The discussion includes the uptake of the research by other researchers, industry partners, and the CPPS engineering and SPL communities. Furthermore, the section presents future work based on the limitations of the approaches and opportunities for research.

Chapter 5 presents the ten core publications contributing to this cumulative thesis, grouped by the research lines. Furthermore, is presents a table with additional publications composed throughout this thesis.

CHAPTER 2

# Synopsis of Publications

This chapter provides the synopsis of the ten publications that form this cumulative thesis and their relation. Table 2.1 lists these publications chronologically with their reference number, title, and year. Furthermore, it denotes the author's position and the research questions to which the particular publication contributes.

Throughout this thesis, we approached the issue of providing means and methods for explicit knowledge modeling and advanced applications for knowledge modeling for CPPSs from several perspectives. Furthermore, we focused on efficient reuse and variability management for multidisciplinary engineering of CPPS families in particular. The synopsis of the publications is grouped into the three main themes of the thesis: PKMO, ACEA, and IRVM.

Additionally, Table 2.2 shows a different view on the core publications regarding the research lines, design artifacts, and targeted VDI 3695 *measures* and SPL *capabilities*. The table lists the research lines mapped to the design artifacts in the first and second columns. Artifact **A0.0** represents the overall idea of the thesis contributions supported by two publications. The third column shows the VDI 3695 *measures* and SPL *capabilities* that the design artifact supports. The last column shows the core thesis references that contributed to the particular research line and design artifact.

## 2.1 Production Knowledge Models

This section summarizes the author's relevant publications for this thesis regarding the overall idea of integrated variability management for CPPS engineering. First, we utilized the standard VDI 3682 model for PPR aspects introducing the variability management. Furthermore, we suggest a superimposed model and the corresponding architecture for an industrial information system that supports this integrated approach. Based on this

| Ref# | Title | Venue | Year | Author | **RQ1**. | **RQ2**. | **RQ3**. |
|---|---|---|---|---|---|---|---|
| #01 [118] | "Towards modeling variability of products, processes and resources in cyber-physical production systems engineering" | SPLC | 2019 | 1st | × | × | × |
| #02 [117] | "Integrating Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering" | SPLC | 2020 | 1st | × | × | × |
| #03 [43] | "Variability Transformation from Industrial Engineering Artifacts: An Example in the Cyber-Physical Production Systems Domain" | SPLC | 2020 | 2nd | × | | × |
| #04 [129] | "Patterns for Reuse in Production Systems Engineering" | SEKE | 2021 | 1st | × | × | |
| #05 [127] | "A reusable set of real-world product line case studies for comparing variability models in research and practice" | SPLC | 2021 | 1st | × | | × |
| #06 [130] | "A Domain-Specific Language for Product-Process-Resource Modeling" | ETFA | 2021 | 1st | × | | × |
| #07 [128] | "Patterns for Reuse in Production Systems Engineering" | IJSEKE | 2021 | 1st | × | × | |
| #08 [132] | "Efficient Production Process Variability Exploration" | VaMoS | 2022 | 1st | | | × |
| #09 [135] | "Organizing reuse for production systems engineering with capabilities and skills" | AT | 2023 | 1st | | × | × |
| #10 [137] | "Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering" | JSS | 2023 | 1st | | | × |

Table 2.1: Chronological list of core publications part of the thesis.

idea, the section further presents a production knowledge model for CPPS engineering and operations based on the VDI 3682 and a DSL.

In the publication **"Towards Modeling Variability of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering"** [118], we conducted the first rigorous domain analysis with an industry partner, a system integrator of automation for high-performance CPPS. This publication identified essential aspects of CPPS engineering that challenge reuse and variability management methods. In particular, we identified (i) the multidisciplinary roles, requirements, models, and dependencies of the CPPS engineering domain, (ii) the heterogeneity of CPPS engineering artifacts, (iii) the limited usability of available tools for reuse and variability management, and (iv) the evolution of the products and the impact on the interdependent CPPS design.

| Research Line | Description | Measures & Capabilities | Reference |
|---|---|---|---|
| | **A0.0** – Overall idea | M1., M2., M3., M4., M5., C1., C2., C3., C4., C5. | [118, 117] |
| PKMO | **A1.1**. – VDI 3682-based *CPPS knowledge model* with variability modeling support | M1., M2., M3., M5., C1., C2., C3., C5. | [118, 117, 130] |
| PKMO | **A1.2**. – Human- & machine-readable *DSL for the CPPS knowledge model* | M1., M2., M3., M4., M5., C1., C2., C3., C4., C5. | [130] |
| ACEA | **A2.1**. – *Model for advanced CPPS engineering applications* with asset dependencies | M1., M2., M3., M5., C1., C2., C5. | [128, 129] |
| ACEA | **A2.2**. – *Integration of C&S-based concepts* into the knowledge model for CPPS asset reuse | M1., M2., M3., M5., C1., C2., C5. | [128, 129] |
| ACEA | **A2.3**. – *CPPS reference model* for reusable engineering design patterns and component reuse | M2., M3., M4., M5., C1., C2., C4., C5. | [128, 129] |
| ACEA | **A2.4**. – *Systematic method for C&S reuse activities* for loose coupling of PPR concepts | M2., M3., M4., M5., C1., C2., C4., C5. | [135] |
| IRVM | **A3.1**. – Model *mapping rules and transformations* to state-of-the-art variability models | M1., M3., M5., C1., C3., C5. | [43] |
| IRVM | **A3.2**. – *Industrial use cases* for empirical evaluation and open science | M1., M3., M4., C1., C3., C4. | [127] |
| IRVM | **A3.3**. – *Integrated approach* for reusing and configuringCPPS assets with process variability | M1., M2., M3., M4., M5., C1., C2., C3., C4., C5. | [132, 137] |
| IRVM | **A3.4**. – *Toolchain for integrated reuse and variability management* approach | M1., M2., M3., M4., M5., C1., C2., C3., C4., C5. | [137] |

Table 2.2: Research lines and artifacts mapped to CPPS *measures* and SPL *capabilities* with core thesis references.

Furthermore, we investigated how to extend the established knowledge model of the industry guideline VDI 3682 [199] that represents the core assets, i.e., PPR, to address these challenges. As a first idea, we proposed (i) a superimposed knowledge model for the PPR approach and (ii) an integrated reuse and variability management approach utilizing orthogonal variability models to the PPR model, (iii) including an architecture draft to model and manage PPR variants. We conceptually evaluated the knowledge model with the *rocker switch* use case (cf. Section 1.1.1) with domain experts from an industry partner. To this end, the publication contributed a feasible knowledge model and an integrated framework for reuse and variability management for CPPS engineering, which we iteratively refined throughout the thesis.

This publication contributes to **RQ1**., **RQ2**., and **RQ3**.. It addresses the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *adaptability* C2., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of multidisciplinary collaboration* C5..

In **"Integrating Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering"** [117], a doctoral symposium publication, we further analyzed the *rocker switch* use case (cf. Section 1.1.1). In this publication, we raised the fundamental challenge of the sequential variability of production processes, which imposes an additional element of complexity. We advanced the knowledge model toward representing production process dependencies in a precedence graph to restrict their potential variability. Furthermore, we argued that current variability modeling approaches treat structural and behavioral variability separately, leading to an inaccurate CPPS model. We also stressed the weak understanding of change impacts in the product portfolio on CPPS production process properties requiring a more agile and iterative process. The publication provided the fundamental insight that combining structural and behavioral variability is necessary. It could only be achieved by integrating different types of variability models to utilize their particular benefits, such as visibility conditions for sequential configuration. Furthermore, the publication sketched a research agenda for planning the artifacts and publications of the thesis.

This publication contributes to **RQ1**., **RQ2**., and **RQ3**. by addressing the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *adaptability* C2., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of multidisciplinary collaboration* C5..

In the publication **"A Domain-Specific Language for Product-Process-Resource Modeling"** [123], we presented an extension of a work-in-progress publication [130]. This extension included the conceptual design for an engineering knowledge model based on the VDI 3682 guideline [199]. The model aimed to represent PPR aspects of CPPSs along with their associated constraints in a comprehensive syntax. To develop this model, we identified requirements in workshops with domain experts from an industry partner in discrete manufacturing. Furthermore, we designed and prototyped a DSL that implements this knowledge model. Additionally, we presented a mechanism to map and evaluate the constraints. This mechanism utilizes advanced SQL queries and supports well-established database technologies used in the industry. We conceptually evaluated the PPR–DSL with the *rocker switch* use case (cf. Section 1.1.1), demonstrating the DSL's capability to represent the functional aspects of CPPS and specify and assess constraints. Furthermore, we measured the performance of the constraints evaluation mechanism using a programmatic approach. Additionally, we conducted a workshop and interviews with domain experts from CPPS engineering for iterative improvements. The knowledge model and DSL were iteratively refined and used in several core publications, as can be seen in Figure 3.1 and multiple further publications.

This publication contributes to **RQ1**. and **RQ3**. by addressing the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *adaptability* C2., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of multidisciplinary collaboration* C5..

## 2.2 Advanced CPPS Engineering Applications

The previous section presented publications concerning the PPR production knowledge model for CPPS engineering and operation. This knowledge model aimed particularly at integrating production processes with the PPR approach and variability modeling. This section discusses publications that cover the advancements of this knowledge model and advanced engineering methods for CPPS engineering based on this knowledge model.

In **"Patterns for Reuse in Production Systems Engineering"** [129], we extended a domain analysis from Meixner et al. [125]. This extension aimed at eliciting requirements for representing PPR assets and their dependencies in an engineering knowledge graph. We introduced the Industry 4.0 Asset Network (I4AN) concept and provided a concrete meta-model for the I4AN. The meta-model included multidisciplinary dependencies between different PPR assets stemming from multiple engineering disciplines. Furthermore, we introduced boundary objects, which help distinguish between closely related assets, their values, and dependencies. These objects allowed us to build a group of linked assets, serving as a foundation for reusable CPPS components, similar to component concepts in software engineering. The asset groups in the reference architecture can expose regularly recurring connected CPPS assets within CPPS engineering. Inspired by design patterns from software engineering, we motivated the implementation of reuse patterns specifically for CPPS engineering. To this end, we presented four basic patterns for reuse engineering. These patterns aimed to improve the creation of concrete patterns that enhance reuse maturity and efficiency. To evaluate the I4AN concept and model, we conducted a conceptual feasibility study with a use case from the domain analysis. The evaluation indicated that the I4AN model satisfies the elicited requirements and enables CPPS engineers to identify patterns for reuse in their disciplines. We further refined the concepts of the publication in [129].

This publication contributes to **RQ1**., **RQ2**. by addressing the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *adaptability* C2., *enhanced quality and consistency* C4., and *facilitation of multidisciplinary collaboration* C5..

The publication **"Patterns for Reuse in Production Systems Engineering"** [128] extended the previous publication [129], sharing the same name, in the "International Journal for Software Engineering and Knowledge Engineering". In this publication, we extended the previously developed meta-model with C&Ss for improved loose coupling between production processes and resources. Furthermore, we refined the dependency representation with technical and functional links from CPPS engineering. Additionally, we added trace links for improved coordination between the engineering disciplines. Finally, we defined a reference model for the CPPSs domain as recommended in the analysis activity of Domain and Application Engineering (DAE) [152]. In the context of Production Systems Engineering (PSE), reference models formally describe engineering knowledge, serving as a common foundation for communication [200]. The reference model provides an abstract representation of engineering elements in the I4AN required to represent typical CPPS designs. We designed and implemented the approach with a prototype based on a well-established graph database that can create and query the I4AN as an engineering graph. Additionally, we examined the approach with the prototype and a feasibility study from the real world. The concepts of the publication were further refined and used in several publications, such as [13, 15, 17, 134, 211], which use the meta-model and reference architecture as a significant building block.

This publication contributes to **RQ1**. and **RQ2**. by addressing the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *adaptability* C2., *enhanced quality and consistency* C4., and *facilitation of multidisciplinary collaboration* C5..

The publication **"Organizing reuse for production systems engineering with capabilities and skills"** [135] picked up our previous approach of reusing PPR models with capabilities [125]. In the publication, we proposed the Capability and Skill Reuse (CSR) framework based on DAE [152]. Therefore, we categorized requirements from a concise literature survey that we conducted [58] into the four groups *capability description*, *capability and skill selection*, *skill implementation time*, and *skill run time*. Based on these categories, we investigated how DAE activities need to be conducted to elicit PPR models and C&Ss from engineering artifacts and models to establish a reuse lifecycle. From this investigation, we proposed the CSR framework and detailed how PPR and C&Ss can be used for every DAE activity. We discussed the CSR framework compared to the traditional approach of CPPS engineering and linked them to categorized benefits from the survey [58].

This publication contributes to **RQ2**. and **RQ3**. by addressing the VDI 3695 measures of *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *adaptability* C2., *enhanced quality and consistency* C4., and *facilitation of multidisciplinary collaboration* C5..

## 2.3 Integrated Reuse and Variability Management for CPPS Engineering

The prior sections have introduced several publications covering the integrated reuse and variability management approach. In particular, these are Meixner et al. [118] and Meixner [117] that identified fundamental challenges and proposed extending the VDI 3682 knowledge model [199]. Furthermore these publications motivated combining different variability models to use their particular strengths, for instance, using decision models for production process sequence configuration. This section presents the publications that cover primarily integrated reuse and variability management approaches.

In **"Variability Transformation from Industrial Engineering Artifacts: An Example in the Cyber-Physical Production Systems Domain"** [43], we stressed the need for model transformations from engineering artifacts with variability to well-established variability models. This method aimed to address the challenge of scattered and implicit knowledge distributed across various disciplines. To address this requirement, we introduced the Variability Evolution Roundtrip Transformation (VERT) process for iterative round-trip transformation process for custom-developed engineering variability artifacts. The approach allows engineers to transform such engineering artifacts to a feature model, evolve and optimize the model and transform it back to the original engineering artifacts. Building on a recently developed transformation approach, we developed model mappings from PPR model representations to feature models. We evaluated the feasibility of the process using a real-world use case from an industry partner. To strengthen our study's rigor and further refine our approaches, we interviewed domain experts and reflect on lessons learned regarding our approach.

This publication contributes to **RQ1**. and **RQ3**. by addressing the VDI 3695 measures of *models and description languages* M1., *re-use* M3., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *variability management* C3., and *facilitation of multidisciplinary collaboration* C5..

The publication **"A reusable set of real-world product line case studies for comparing variability models in research and practice"** [127] aimed at comparable evaluations and empirical studies for variability modeling for CPPS engineering. To this end, the publication presented four iteratively developed reusable real-world case studies describing variability in CPPSs, which are otherwise often not accessible.[1] The use cases were first modeled in the PPR–DSL, previously developed in Meixner et al. [123]. Then, we automatically transformed these artifacts to well-known variability models, for example, the Product Feature Models (FM), with TRAVART transformation operations [46], based on the previously developed transformations [43]. We evaluated the completeness and expressiveness of the transformed variability models compared to the PPR models. The results indicated that the models' content and variability can be fully transformed into feature models. The four use case studies were added to the ESPLA catalog, a collaborative catalog of case studies for software product line adoption.[5] Furthermore, the case studies were used in several successive publications as a foundation for evaluation, such as [132, 137].

This publication contributes to **RQ1**. and **RQ3**. by addressing the VDI 3695 measures of *models and description languages* M1., *re-use* M3., *quality assurance* M4., and the SPL

[1]GitHub Repository: `https://github.com/tuw-qse/cpps-var-case-studies`

capabilities *efficient reuse* C1., *variability management* C3., and *enhanced quality and consistency* C4..

In **"Efficient Production Process Variability Exploration"** [132], we presented the Iterative Process Sequence Exploration (IPSE) approach. The approach comprises a process that captures CPPS DAE knowledge in PPR–DSL models and transforms them to a structural Product FM and behavioral production Process DM. Furthermore, the approach establishes a configuration process that reduces the configuration space of the production processes based on the product configuration. In subsequent steps, the approach allows the exploration of the production process sequences in a decision model. For production process sequence design, the approach is the first to combine configurable structural and behavioral variability required for CPPS design. The approach was complemented with a prototype based on state-of-the-art technology. We evaluated the approach in the feasibility study on a manufacturing work line from automotive production and compared it with the traditional approach of process sequence planning.

This publication contributes to **RQ3**. by addressing the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *adaptability* C2., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of multidisciplinary collaboration* C5..

In the publication **"Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering"** [137], we extend our work [132] towards (i) additional process steps for production resource definition in the PPR–DSL, (ii) transformations to a Resource FM, (iii) its reduction and configuration, (iv) the generation of CPPS artifacts, (v) and feedback loops for iterative CPPS design. Furthermore, we improved the prototype to evaluate the feasibility of the approach, extending it with (i) concrete transformations of the PPR–DSL into a Resource FM and Cross-Discipline Constraints (CDCs) and to elicit the dependencies from the different variability models, (ii) a novel DM editor for modeling and configuring DOPLER decision models, that in this context allows representing and configuring production process sequences, (iii) an automated reduction of the possible production process DM configuration based on the product configuration to lower the complexity of production process sequence configuration (iv) support for resource configuration, and (v) IEC 61499 artifact generation to automate the creation of control software based on the PPR configuration. To empirically evaluate the approach, we investigated the approach's feasibility of selecting adequate CPPS variants in the four previously developed case studies [127]. We also applied the approach in a new case study with engineers from heterogeneous backgrounds. The publication presents the first exploration of the combined utilization of feature and decision models for configuring CPPSs and creating CPPS artifacts.

This publication contributes to **RQ3**. by addressing the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *adaptability* C2., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of multidisciplinary collaboration* C5..

### 2.3.1 Summary: Synopsis of Publications

The publications contributing to this cumulative thesis are grouped into the three main research lines (i) *Production Knowledge Models*, (ii) *Advanced CPPS Engineering Applications*, and (iii) *Integrated Reuse and Variability Management for CPPS engineering*. The ten publications comprising this cumulative thesis support the research questions in the following way (cf. Table 2.1).

Two publications [117, 118] introduced the overall ideas of the thesis shaping the research questions (**RQ1**.–**RQ3**.) and presenting first ideas. In addition to those publications, five publications address **RQ1**.. Three publications, specifically, develop, extend, and evaluate the PPR production knowledge model with variability and establishing a corresponding DSL [43, 127, 130]. Two publications advance the knowledge model for coordination and communication towards engineering patterns in an I4AN [128, 129].

Beyond the two initial publications, three publications address **RQ2**.. Two publications develop the I4AN reference model for identifying engineering design patterns [128, 129]. One publication supports the research question with a systematic approach for eliciting and developing reusable PPR models supported by C&Ss [135].

On top of the two initial publications, six publications address **RQ3**.. One publication investigated transforming industrial engineering artifacts with variability to state-of-the-art variability models to gather insights on product and process variability [43]. One publication extended the VDI 3682 knowledge model towards a knowledge model for PPR variability modeling in a textual DSL with constraint evaluation [130]. One publication established the transformation of the PPR–DSL to state-of-the-art variability models, including product and resource feature models and process decision models [43]. Two publications comprise the IPSE and Extended Iterative Process Sequence Exploration (EIPSE) approach [132, 137] for model transformations from the PPR model to feature and decision models. Furthermore, it allows a reduction of subsequent configurations of a Product FM, a Process DM, and a Resource FM. Additionally, it includes the development of a decision model editor and a generation of parameterized IEC 61499 control code artifacts, with a rigorous empirical evaluation.

The publications of this thesis provide methodologies, models, and tools for efficient reuse and variability management for CPPSs families.

These approaches allow explicit production knowledge modeling via the PPR knowledge model and PPR–DSL. This includes the inherent variability of products, production processes, and production resources to address the different dimensions of CPPSs families.

Furthermore, the approaches enable knowledge exchange via engineering design patterns and systematic reuse of engineering artifacts. This enables creating parameterizable and reusable artifacts that foster DAE in the CPPS lifecycle supporting CPPS families.

Engineering artifacts from prior projects can be elicited, developed toward PPR models loosely coupled with C&Ss, and validated using the CSR approach.

The integrated reuse and variability management approach for CPPS engineering allows the modeling of CPPS knowledge and allows the subsequent configuration of interdependent products, processes, and resources. This is based on transformations from the CPPS models to feature and decision models and includes, as a showcase, engineering artifact generation with resource control code. The empirical evaluation of the approach in a lab user study with domain experts indicates that the approach lowers engineering expert effort. It also creates suitably reusable engineering models for improved reproducibility and handover to other engineering disciplines.

CHAPTER 3

# Scientific Contributions

This chapter presents the scientific contributions of the thesis project. The contributions forming the core of this thesis project are grouped along the three main research lines introduced in Chapter 1. First, the research line Production Knowledge Models (PKMO) provides foundational knowledge models and techniques to make implicit multidisciplinary production engineering knowledge with variability explicit. Then, the research line Advanced CPPS Engineering Applications (ACEA) describes methods and models for engineers to reuse multidisciplinary production engineering knowledge systematically with reference models and design patterns. Finally, the research line Integrated Reuse and Variability Management (IRVM) integrates SPL techniques with CPPS engineering to manage the variability of CPPS families. This research line particularly considers production process sequences and maintaining the separation of concerns from the different engineering disciplines.

For better visualization, Figure 3.1 depicts the publications and their relation over time using dashed blue arrows for PKMO, dotted red arrows for ACEA, and densely dotted violent arrows for IRVM. The following sections describe the main scientific contributions to these themes.

## 3.1 Production Knowledge Models

Externalizing implicit domain knowledge on CPPSs is essential for reusing CPPS design and engineering artifacts. Therefore, *PKMO* that structure the domain knowledge are crucial. This section first introduces the superimposed PPR model, based on the VDI 3682 model [199] notation, for visually representing PPR variability. Subsequently, the section presents a production knowledge model and a DSL for PPR [181] modeling based on the underspecified VDI 3682 model [199].

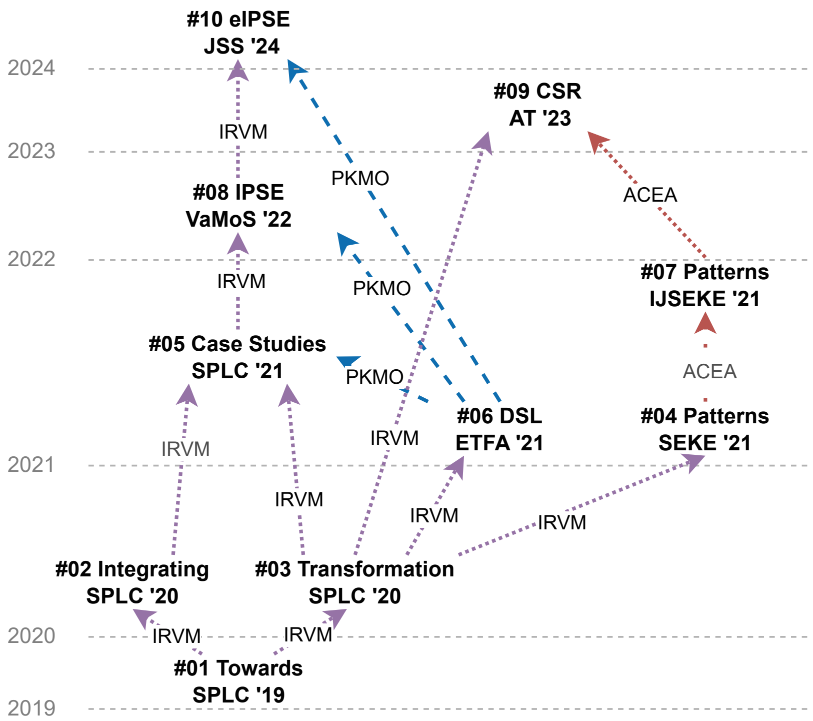


Figure 3.1: The publications forming this cumulative thesis' core (cf. Table 2.1) follow the three research lines: PKMO, ACEA, and IRVM approaches (cf. Chapter 1). Their relations are highlighted by specific arrows labeled with a research line.

### 3.1.1 Superimposed PPR Modeling

Models apply three core principles, which are the abstraction and reduction of the original, and pragmatism in building those models [189]. We postulate that models are an abstraction of the world restricted by a domain that reduces the scope of the world to model. In the context of this thesis, we understand knowledge models as abstract representations of (engineering) knowledge for a domain-specific context. The context reduces the scope of knowledge subsequently restricting the model.

The VDI 3682 [199] provides a visual and formal guideline for modeling PPR models as production process descriptions. While the formal guideline details particular concepts, its explanatory notes get increasingly imprecise. Therefore, Part 3 of the guideline aimed at an XML representation of the knowledge model. However, this part is not expected before 2025, posing a significant limitation.

Unfortunately, such PPR models also do not allow modeling the variability of the PPR aspects. CPPS engineering practice often uses a so-called *type representative* [70] that expresses not only a single product but a set of product features, representing several of the most important types, in a "virtual" product, i.e., a 150% model [194, 107]. The

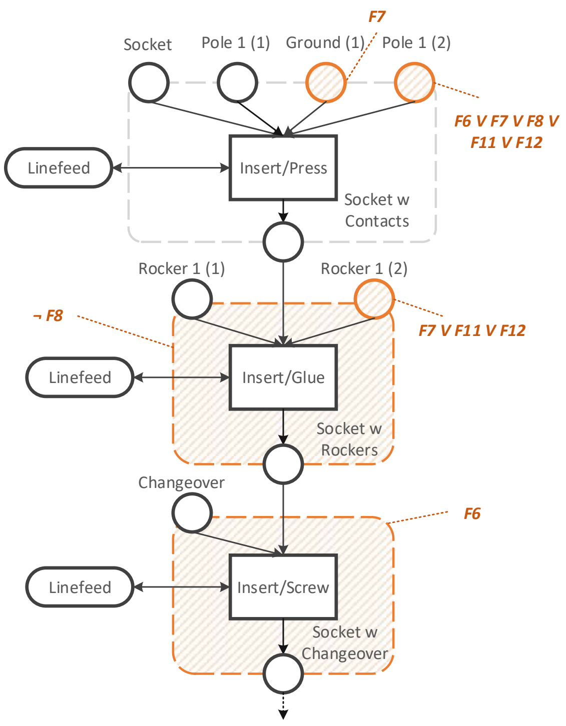


Figure 3.2: Concept of a superimposed model for PPR models, in adapted VDI 3682 notation [199].

engineers then utilize this type representative to plan potential production processes and resources. This planning is done using TCMs and selecting products covering a particular set of product parts. According to our industry partner, another strategy in CPPS engineering is to test variable products by taking several "real" products as representatives and simulating their production one after another.

We picked up the VDI guideline and introduced *superimposed PPR models* in Meixner et al. [118] as type of 150% PPR model. This notation aimed to support engineers allowing them to visually express the variability in PPR models. Figure 3.2 shows such an superimposed model for a part of the *rocker switch* use case (cf. Section 1.1.1). PPR elements in black are common model parts shared by every product and production process. For instance, all products share the parts *Socket*, *Pole 1 (1)*, *Rocker 1 (1)*, and *Changeover*. The part *Ground (1)* is only used in product *F7*. Similarly, the last production step of the production process model is only used for products *F6*. The product annotations are concatenated with OR if several products use one of the PPR elements. For example, *Pole 1 (2)* is used for the product types *F6*, *F7*, *F8*, *F11* and *F9*, while the second production step is not used in product type *F8*.

We evaluated the approach in the feasibility study with the *rocker switch* use case. Then we discussed the utility of the superimposed PPR model with domain experts from our industry partner. The experts found the superimposed model with the product variants embedded in the production processes to improve their way of modeling compared to TCMs [118]. However, the domain experts also noted that it requires tool support to maintain the potentially complex model [118]. To this end, we experienced that the superimposed PPR model can grow quite large and get cluttered, for instance, with annotations, in use cases of medium size. We argue that such models have to be designed in a modular way, which can counter this issue, as the VDI 3682 proposes. However, this argument needs to be investigated in additional research.

*To summarize*, superimposed PPR models visually capture the variability of functional PPR aspects and offer an easy-to-understand variability representation for CPPS engineering. This way, it enhances the established PPR approach [181, 199] and helps engineers to better communicate their engineering knowledge with the variability. However, the rising complexity of the superimposed PPR models requires additional tool support.

**Research Outcome.** In research, Fidan et al. [53] adopted this notation to represent variability in production processes for frugal production. The aim is to model the full range of the production processes in the superimposed model. The model is then tailored to local needs in developing countries, thereby simplifying complexity and supporting sustainable production regarding the Sustainable Development Goals (SDGs) 6, 9, and 12.[1]

### 3.1.2 PPR Production Knowledge Model and PPR–DSL

Picking up the superimposed PPR model, we require a concrete, comprehensive model to effectively manage the PPR aspects and their variability inherent in CPPS engineering. The following section introduces the *Production Knowledge Model* for PPR with variability as the foundation for knowledge externalization in the *PPR–DSL*.

**PPR Production Knowledge Model** Based on the VDI 3682 guideline's limitation to provide a machine-readable and precise knowledge model and the positive perception of the superimposed PPR model, we designed a concrete knowledge model. This model discards elements with largely imprecise explanations, creating a pragmatic knowledge model for CPPS engineering in Meixner et al. [123] and Meixner et al. [130]. Figure 3.3 and Figure 3.4 show the meta-model (elements) of this knowledge model.[2]

Figure 3.3 shows the main entities of the meta-model in an UML class diagram – Product, Process, and Resource, derived from the Vdi3682Object, and the AssemblySequence that serves as a container for the model. To uniquely identify the entity, the Vdi3682Object is associated with an Identification. The class has a Boolean value isAbstract, allowing abstraction and inheritance. Furthermore, it has a Boolean value isComposed, indicating whether the particular entity is composed of entities of similar type and a list of Characteristics that characterize the entity. The Identification refers to identifyingAttributes that consist of

[1]UN SDGs: `https://sdgs.un.org/goals`

[2]Note that the meta-model here is located on level *M2* in the MOF.

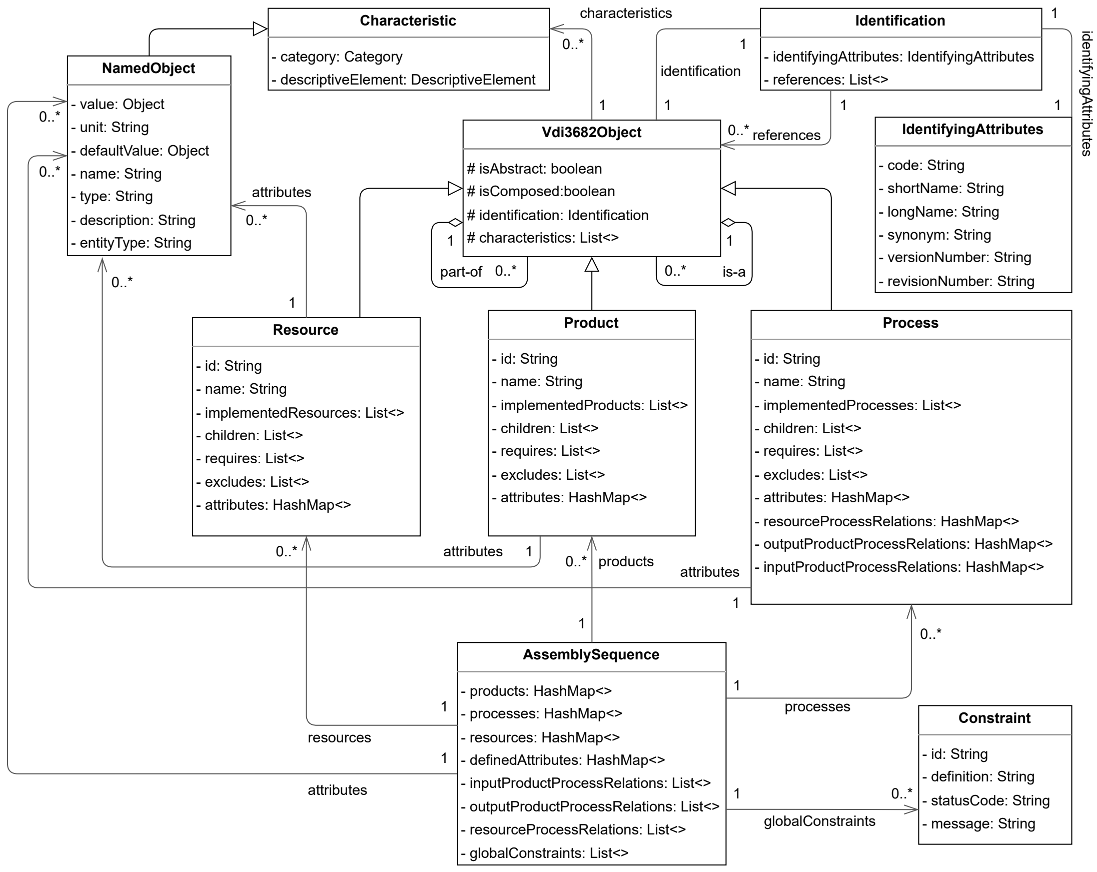


Figure 3.3: The core entities *Product*, *Process*, and *Resource* and their inheritance hierarchy of the PPR knowledge model as UML class diagram.

several values, such as a code, versionNumber, and revisionNumber, that identify an entity. Furthermore, the Identification can also refer to other Vdi3682Objects to describe close relations. Characteristics a Category, such as a type of value, and an DescriptiveElement. Additionally, Characteristics have minor properties that we stripped from the model for better understandability and, thus, are not shown in the figure. The AssemblySequence holds the Products, Processes, and Resources of the model. Furthermore, it contains the definition of the attributes that can be reused in the particular PPR entities. The AssemblySequence also has lists of the relations between the PPR entities, which are explained in the paragraphs below. Finally, the AssemblySequence holds a list of globalConstraints that are used to define relations between the PPR entities that can be evaluated. These constraints can be compared to constraints in the variability models from SPL engineering. PPR elements have a list of NamedObjects, derived from Characteristics, that represent a particular element's attributes. These specialize the characteristics of the Vdi3682Object. Each of the three PPR entities has several lists holding instances of its own type, explained in the following utilizing the Process entity. However, note that the relations of these associations are not shown in the figure to keep it de-cluttered. The Process holds a list of abstract entities that it is derived from, cf. Figure 3.3 implementedXXX, for instance, implementedProcesses. For instance, for the *shift fork* use case (cf. also Figure 1.5 in Section 1.1.1), the *Pipe* is an abstract concept as a parent, with *Pipe2*, *Pipe3*, and *Pipe8* as concrete concepts, referring to *Pipe* in the implementedProducts. Furthermore, the entities hold a list of children that represent particular products they comprise, accumulating throughout the production process. For example, the finished shift fork includes all the required product parts as children. The following two lists concern variability modeling and management. The requires list holds required entities for this entity to be produced, for instance, required products to which the specific product is mounted. Similarly, PPR entities have a list of excludes that they cannot be produced with. For instance, in the *shift fork* use case *Pipe2*, *Pipe3*, and *Pipe3* exclude each other. These lists can be compared to implication and exclusion criteria from variability modeling. Additionally to the Product and Resource entities, the Processes has structures for relations to Resources, resourceProcessRelations, and Products that are inputs and outputs of the production process, inputProductProcessRelations and outputProductProcessRelations. We detail this in the paragraphs below.

We introduced concrete relations for the domain model to represent the relations in the PPR models. Figure 3.4 shows these relations, which are inherited from an entity Relation (cf. Figure 3.6). InputProductProcessRelations model the relation between a particular *input* Product and production Process. This relation also maintains a reference to the OutputProductProcessRelation to indicate from which production Process the *input* Product comes. This is needed to model abstract Products as a blueprint for reusable PPR models. The OutputProductProcessRelation maintains references to a particular *output* Product and production Process. To model relations between a concrete production Process and Resource, we introduced the ResourceProcessRelation with corresponding references. The AssemblySequence maintains global lists of the relations between Products, Processes, and Resources. Products and Processes are related to each other via InputProductProcessRelations and OutputProductProcessRelations. In the same way, Processes and Resources are connected via ResourceProcessRelation.

*To summarize*, due to limitations of the VDI 3682 to provide a precise, machine-readable knowledge model, we extracted a pragmatic knowledge model tailored for CPPS engineering for the PPR concepts. The knowledge model supports modeling variability through abstraction and inheritance between elements and lists of required and excluded elements. The knowledge model also embeds advanced constraints that cannot be expressed with the basic functionality. This refined model addresses the imprecise elements of the original guideline and offers an approach to capture, manage, and communicate engineering CPPS knowledge.

**Product-Process-Resource Domain-Specific Language** DSLs are languages tailored to a domain intended to be easily written and understood by the experts of that domain [56]. We designed the extensible PPR–DSL [130, 123] based on the above-

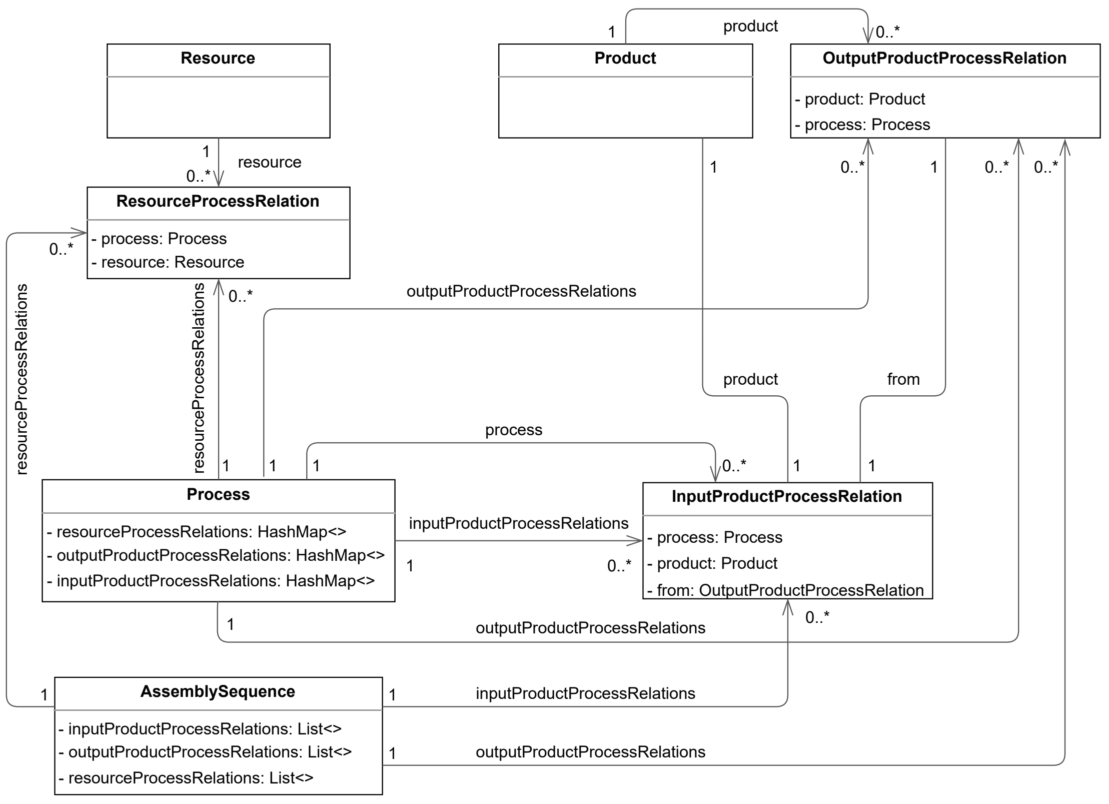

Resource
Product
OutputProductProcessRelation
- product: Product
- process: Process
product
resource
ResourceProcessRelation
- process: Process
- resource: Resource
resourceProcessRelations
outputProductProcessRelations
product
from
process
Process
- resourceProcessRelations: HashMap<>
- outputProductProcessRelations: HashMap<>
- inputProductProcessRelations: HashMap<>
inputProductProcessRelations
InputProductProcessRelation
- process: Process
- product: Product
- from: OutputProductProcessRelation
AssemblySequence
- inputProductProcessRelations: List<>
- outputProductProcessRelations: List<>
- resourceProcessRelations: List<>
1
0..*

Figure 3.4: Combined PPR knowledge model as UML class diagram.

mentioned knowledge model. The DSL comprises a custom-tailored mapping of the knowledge model to DSL elements for CPPS engineers. This alignment helps them relate to their understanding of the concepts, enabling quick comprehension and writing of the PPR–DSL.

Listing 1 and Listing 2 show two listings in PPR–DSL for the shift fork model (cf. Section 1.1.1). Listing 1 defines first two Attributes uniquely identified by their ID *length* in line 1 and *partialProduct* in line 7. Furthermore, it defines *length*, as numerical value, and *partialProduct*, as Boolean value, with their defaultValues, types, and units. Line 12 shows an abstract Product *Fork*, keyword isAbstract, that sets the Attribute *partialProduct* to true, indicating that this concept is a part of the product. Lines 15 to 20 define a concrete partial Product *Fork3* with a *length* of 30 millimeters that implements the abstract *Fork*. Additionally, *Fork3* directly requires the Product *Fork5* to be manufactured. Lines 25 to 29 define a partial Product *Pipe2* that excludes the Products *Pipe3* and *Pipe8*, which means they cannot be used in the same product variant. Lines 31 to 34 define an abstract final Product *ForkProduct* that requires, besides others an abstract *Pipe* and *Lock*. *Fork_13* implements the *ForkProduct* and details the abstract required products to *Pipe8* and *Lock3* and further requires an additional *Barrel1_2*.

```
1   Attribute "length": {
2     type: "Number",
3     unit: "mm",
4     defaultValue: 1.0
5   }
6
7   Attribute "partialProduct": {
8     type: "String",
9     defaultValue: "false"
10  }
11
12  Product "Fork": { name: "Abstract Fork",
13    isAbstract: true, partialProduct: "true"
14  }
15  Product "Fork3": { name: "Fork 3",
16    length: 30,
17    implements: [ "Fork" ],
18    requires: ["Fork5"],
19    partialProduct: "true"
20  }
21
22  Product "Pipe": { name: "Abstract Pipe",
23    isAbstract: true, partialProduct: "true"
24  }
25  Product "Pipe2": { name: "Pipe 2",
26    implements: ["Pipe"],
27    excludes: [ "Pipe3", "Pipe8" ],
28    partialProduct: "true"
29  }
30
31  Product "ForkProduct": { name: "ForkProduct",
32    isAbstract: true,
33    requires: [ "Barrel1_1", [...], "Fork3", "Pipe", "Lock"],
34  }
35  Product "Fork_13": { name: "Fork 13",
36    implements: ["ForkProduct"],
37    requires: ["Pipe8", "Lock3", "Barrel1_2"]
38  }
39
40  Resource "Linefeeds": { name: "Linefeeds", isAbstract: true }
41  Resource "LF_3": { name: "LF_3", implements: [ "Linefeeds" ] }
42  Resource "LF_4": { name: "LF_4", implements: [ "Linefeeds" ] }
```

Listing 1: Excerpt of the products and production resources of a PPR model for the *shift fork* case study in PPR–DSL [131].

Listing 1 also shows simplified Resource definitions of abstract *Linefeeds* that can feed workpieces to the production system, where *LF_3* is capable of feeding larger pieces to the system than *LF_4* (lines 41 to 42). As both Resources can be used for the configuration of a CPPS, they do not exclude each other. Using the inheritance structure, the model can build a hierarchical family of Resources that engineers can reuse across different engineering projects and even build catalogs similar to *EClass* or the *technical library* that our industry partner currently modeled in Excel.

Listing 2 shows the definition of production Processes in the PPR–DSL. Lines 1 to 5 define a Process *InsertFork3* that inserts the input product *Fork3* to the production system, inputs keyword. The outputs contain an abstract Product with the productId *Forkproduct* and the output operation ID *OP17* to identify the output product with the particular production process uniquely. The Process also defines resources with the resourceId *Linefeeds* that can later then be configured with a concrete Resource, for example, the *LF_3*. Lines 7 to 13 define a Process *WeldFork3* that requires the processes *InsertFork3* and *WeldFork5* to be completed before it can start. It also requires that the input Products a particular *Pipe* and the concrete Products *Fork3* and *Fork5* are present in the system. Similar to the previous Process, the output is a particular *Forkproduct*, and the Process uses a Resource from the family of *LaserWeldingRobots*.

```
1   Process "InsertFork3": { name: "InsertFork3",
2     inputs: [ {productId: "Fork3"} ],
3     outputs: [ {OP17: {productId: "ForkProduct"}}],
4     resources: [ {resourceId: "Linefeeds"} ]
5   }
6
7   Process "WeldFork3": { name: "WeldFork3",
8     requires: [ "InsertFork3", "WeldFork5" ],
9     inputs: [ {productId: "Fork3"}, {productId: "Pipe"},
10      {productId: "Fork5"} ],
11    outputs: [ {OP18: {productId: "ForkProduct"}}],
12    resources: [ { resourceId: "LaserWeldingRobots" } ]
13  }
14
15  Constraint "Constraint1": {
16    definition: "Lock1, Pipe2, Pipe3 -> Lock1 implies Pipe2 or Pipe3"
17  }
18
19  Constraint "Constraint2": {
20    definition: "Lock2, Pipe3 -> Lock2 implies Pipe3"
21  }
22
23  Constraint "Constraint3": {
24    definition: "InsertLock1, InsertLock2, InsertLock3 -> InsertLock1 implies (not
      ↪  InsertLock2 and not InsertLock3)"
25  }
26
27  Constraint "Constraint4":{
28    definition: "Pipe descendants all -> all.length > 10"
29  }
```

Listing 2: Excerpt of a PPR model for the *shift fork* case study in PPR–DSL [131].

Lines 15 to 25 of Listing 2 show three Constraints that cannot be expressed via the more simple requires or excludes definitions. Therefore, we designed a reduced syntax for PPR models, particularly circumventing the complexity of other constraint evaluation languages, such as OCL.[3] Constraints can be specified *explicitly* or *implicitly*. The Constraint1 expresses that if Product *Lock1* is used that either Product *Pipe2* or *Pipe3* need to be used. Similarly, Constraint2 expresses that if *Lock2* is used *Pipe3* also needs to be used. *Constraint3* defines that if Process *InsertLock1* is used, then Processes *InsertLock2* and

[3]OMG OCL: `http://www.omg.org/spec/OCL/`

*InsertLock3* must not be used. These Constraints explicitly denote the PPR entities. However, we can also specify Constraints *implicitly* by indirectly referring to them via particular manipulators. Therefore, we designed the keyword subtype (for *implements* relations) and descendants (for *containment* relations) in the PPR models followed by a qualifier in the constraint definition. Constraint4 uses the keyword descendants and qualifier all. In Meixner et al. [130], we list additional keywords and functions that we support in the constraint syntax.

To validate the PPR knowledge model and, in particular, their constraints, we aimed to utilize state-of-the-art technology widely available in industrial and manufacturing contexts for broader acceptance. Therefore, we first implemented the meta-model as an interface library in Java. Then, we implemented the PPR–DSL in a Java library, a parser to read DSL files, and a corresponding DSL writer. Furthermore, we investigated the Structured Query Language (SQL) as a well-established technology in the industry, which provides a standardized syntax for querying, in this case, relational data models. We mapped the PPR–DSL constraint syntax to SQL query templates whose parameters are then replaced at runtime by the concrete values. These queries can be quite complex, which means that a particular technology must support, for instance, recursive queries. Therefore, we utilized *PostgreSQL*[4] as a proof-of-concept database to execute these SQL queries and return potential constraint violations. This strategy also goes along with the existing software ecosystem of our industry partner, who did not want to integrate additional technologies such as the Eclipse Modeling Framework (EMF) [190].

We initially evaluated the approach with the *rocker switch* use case by reverse engineering existing artifacts and plans from our industry partner. Throughout the thesis and the later works [127, 137], we employed the PPR–DSL for modeling the PPR aspects of various CPPSs. Furthermore, we gathered feedback from the engineers of our industry partner in a workshop with interviews. Compared to drawing the PPR models, they found the PPR–DSL easier to create and maintain and the structured constraints with their automated evaluation useful, compared to their natural language constraints [130]. Nevertheless, they noted that the PPR–DSL is verbose and emphasized the need for simplification [130]. Furthermore, they confirmed that the implementation lowers the entry barrier to their ecosystem[130].

In ongoing work, we simplified the syntax of the PPR–DSL to address these limitations. For instance, we raised the Boolean attribute isAbstract and the Attribute partialProduct to a prefix of the PPR concepts. Furthermore, we de-cluttered the syntax from superficial quotes and parentheses. This simplification results, for instance, in code like abstract partial Product Fork {...}. Examples of the adopted simplified syntax can be found in Rinker et al. [170] and Rinker et al. [172]. To broaden the PPR–DSL to the scientific community and industrial practitioners, in ongoing work, we use Eclipse Xtext,[5] a tooling framework for developing DSLs, to create an improved grammar. We translate this Eclipse Xtext grammar directly to a parser and PPR–DSL editor with syntax highlighting and support.

[4]PostgreSQL: `https://www.postgresql.org`
[5]Eclipse Xtext: `https://eclipse.dev/Xtext/`

As Eclipse Xtext uses ANTLR,[6] a parser generator, in the background to generate the syntax and parser, we can adapt this ANTLR grammar to use it with different other DSL tool suites, such as the popular Language Server Protocol (LSP) that supports multiple other editors like VSCode.[7]

*To summarize*, the PPR–DSL manifests the previously introduced PPR production knowledge model. This way, engineers can relatively straightforwardly describe PPR models, for instance, in a text editor or collaborative document. Furthermore, we developed a basic embedded constraint language that facilitates expressing relations between PPR elements, such as $x < y$, calculating sums, or more complex variability constraints. This constraint language was mapped to SQL and implemented in PostgreSQL as a widely used industry standard.

**Research Outcome.** The PPR production knowledge model and the PPR–DSL allow to externalize previously implicit knowledge of CPPS engineers in a structured machine-interpretable way. Thereby, they provide the foundation for multidisciplinary knowledge transfer, reusable engineering artifacts, and transformation of the PPR knowledge with variability to well-established variability models. In research, the approach was picked up as a foundation for (i) further research in our group in the context of a PhD project [169, 172], (ii) a novel CPPS engineering information ecosystem by our industry partner at the CDL SQI, (iii) an evaluation of the maturity of DSL ecosystems for CPPSs in a transnational research collaboration [64]. Furthermore, the approach was picked up as a foundation for a visual representation of the PPR models in a line of master theses [35, 153, 100, 21]. However, the visual representation in combination with the PPR–DSL is not in the scope of this thesis.

## 3.2 Advanced CPPS Engineering Applications

The knowledge model presented in the previous section outlined a concrete formal representation for externalized engineering knowledge based on the PPR concepts as primary aspects of CPPSs. The *ACEA* build on this knowledge model to further structure the externalized engineering knowledge. They also aim to enhance communication with others via design patterns and systematically reuse these patterns and engineering artifacts.

The section first presents the I4AN knowledge model to structure engineering knowledge further along the engineering disciplines and CPPS aspects. Second, it introduces the I4AN reference model that helps organize engineering knowledge as a foundation for reusable engineering design patterns. Finally, the section describes how to systematize the elicitation and reuse of engineering artifacts employing C&Ss for decoupled processes and resources.

[6]ANTLR: `https://www.antlr.org`
[7]VSCode: `https://code.visualstudio.com/`

### 3.2.1 I4AN Meta-Model and Reference Model for Engineering Design Patterns

Design patterns are common solutions for recurring problems that serve as a basis for coordination and communication [57]. Similarly, the VDI 3695 recognizes a *reference model* as a formal description of engineering knowledge forming a common basis for communication [200]. Therefore, we consider engineering design patterns for CPPSs essential, especially in multidisciplinary CPPS engineering.

To this end, we extended our knowledge model (cf. Section 3.1.2) to the I4AN model in Meixner et al. [128, 129]. This extension introduced multidisciplinary dependencies to link engineering assets and artifacts across disciplines. It also aims to define distinct concept delimitations to support engineers in identifying such patterns. Furthermore, we developed a reference model, in the sense of the VDI 3695 reference models [200], for structuring engineering knowledge. This knowledge model builds the foundation for eliciting engineering design patterns and conceptual reuse in CPPS engineering.

**I4AN Knowlede Model** The following detailed description of the I4AN meta-model, shown in Figure 3.5 as a UML class diagram, explains the essential elements, attributes, and relations.

The I4.0Asset, as material or immaterial good with a digital representation in an engineering organization[8] [151] shown in the middle of Figure 3.5, builds the root of the knowledge management model. We carry over several of the Vdi3682Object characteristics to the I4.0Asset. To this end, an I4.0Asset can be a specialization of another I4.0Asset via the *is-a* relation or build a hierarchy via the *part-of* relation. Furthermore, the I4.0Asset comprises a list of attributes of type NamedObject. We extended the class NamedObject from the previous model with a part-of relation. This relation enables a hierarchical structuring of attributes that follows the Asset Administration Shell (AAS) design [151] towards a standardized representation of attributes views for multidisciplinary CPPS engineering. We apply these measures as information modeling at various levels of system functionality is essential in CPPS engineering [97]. Derived from the I4.0Asset, we have the three ubiquitous PPR elements [181] and the supplementary Capability concept [149].[9] The original PPR approach [181] and the VDI 3682 [199] establish a strict connection between processes and resources based on a direct usage in the original model. This tight coupling between processes and resources creates a relatively inflexible model [97]. To better decouple processes and resources, Keddis et al. [93] and Pfrommer et al. [149] and others [58] presented the Capability concept. In this context, a capability is defined as "an implementation-independent specification of a function in industrial

[8] Asset Administration Shell: `https://opcfoundation.org/documents/30270/`

[9] In the original publications [129, 128], the concept was named *Skill*. However, here, we use the already more established term *Capability* for the concept, leaving the *Skill* concept as its concrete implementation by a particular *Resource*. A more recent model of the capability and skill concepts, including production services, was published in a joint publication with this thesis' author as one author in [97].

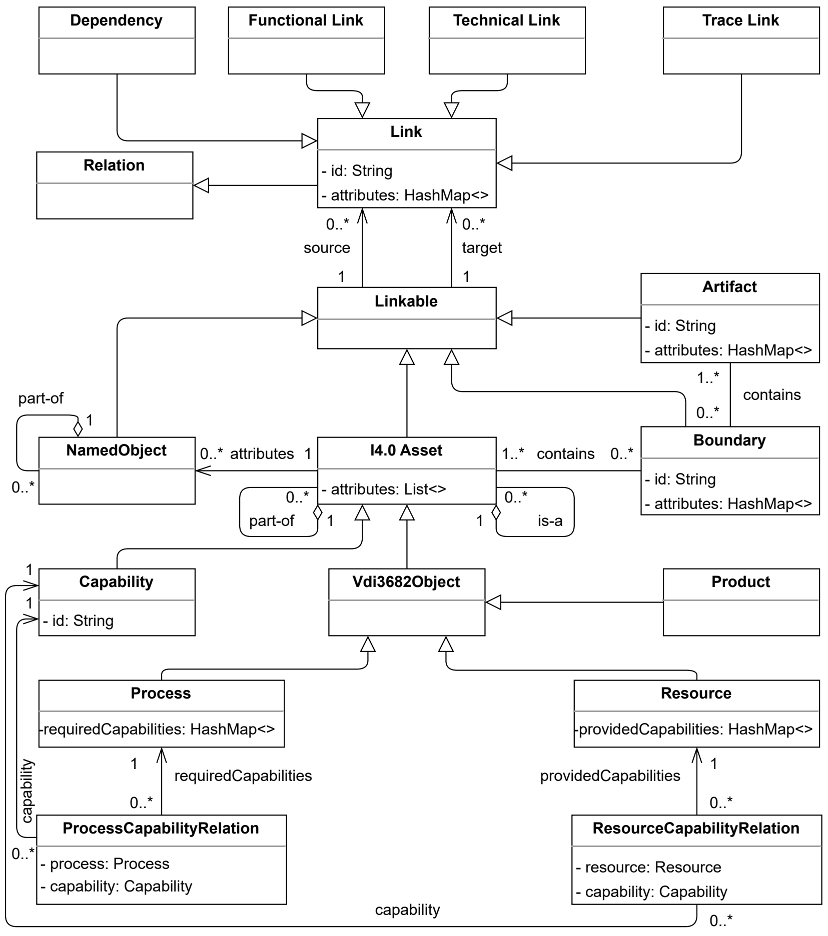


Figure 3.5: Domain model of the I4AN as UML class diagram.

production to achieve an effect in the physical or virtual world," specifying a function of a process [97]. Consequently, the Resource holds providedCapabilities, a list detailing the Capabilities, identified by an ID, provided by a Resource. Similarly, the Process has requiredCapabilities, indicating the Capabilities required for the Process. For completeness, we also depict the Product class.

An Artifact models a CPPS engineering artifact from engineering or operation, such as CAD drawings and robot control code or log data and process measurements.

To relate the CPPS concepts among each, we introduced the Link class in the model. For better understandability, Figure 3.6 shows the inheritance structure of the different Relation classes. The Link has two associations to the Linkable class, which acts as a superclass for classes that can be linked, i.e., the I4.0Asset, the NamedObject, the Artifact, and the Boundary classes. The Link class is subdivided into Technical Links, Functional Links, and Trace Links as well as Dependencies to represent various CPPS relationships. A

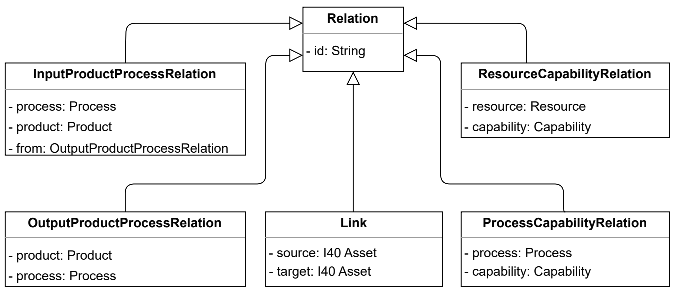


Figure 3.6: *Relation* entity and its inheritance hierarchy of the PPR and I4AN knowledge model as UML class diagram.

*Functional Link* refers to a physical or technical association between assets established by domain experts. It typically involves hardware or direct connections, such as a wired connection from a *Linefeed* to a *Conveyor Belt*, which transmits information between these assets. A *Technical Link* represents a logical or functional connection between assets that work together to perform a specific function in the production process. It focuses on the collaborative roles of the assets to achieve a particular task. For instance, a *Screwdriver* along with its *Bit* and *Controller* collectively provide a screwing function. A *Trace Link* refers to a permanent or temporary association between assets and artifacts or both that are interesting to examine or trace. For instance, it might be interesting within an inspection of an integration test or issue to link several assets and artifacts to a group and inspect their actual measurements along their relations. A Dependency Link is essential for describing patterns where assets or artifacts depend on each other and must be included in a CPPS design when selecting the outgoing concept. For example, in an orchestrated two-robot work cell, an *industrial PC (IPC)* always requires interfaces to the *Robot Controllers*. A Boundary is a container for a group of Assets or Artifacts or both. For instance, it defines the assets associated with a particular pattern. The Boundary also reveals external Dependency Links for a group of assets that extend beyond the Boundary and are intended for reuse.

*To summarize*, the I4AN model extends the previous PPR production knowledge model incorporating dependencies over discipline borders to capture multidisciplinary CPPS engineering knowledge. The model incorporates the PPR and C&S concepts to decouple processes and resources, enhancing system flexibility. Furthermore, it builds the foundation for integrating engineering artifacts from different disciplines. Additionally, the model defines several link types, such as technical and functional, for representing different CPPS engineering relations, beyond others to engineering artifacts and trace links. The model also defines dependencies and boundaries to encapsulate CPPS components, allowing better coordination between the different engineering disciplines.

**I4AN Reference Model and Design Patterns.** To structure engineering knowledge and identify reusable engineering design patterns, we developed the I4AN reference model based on the visual VDI 3682 model. This approach aims to improve communication and

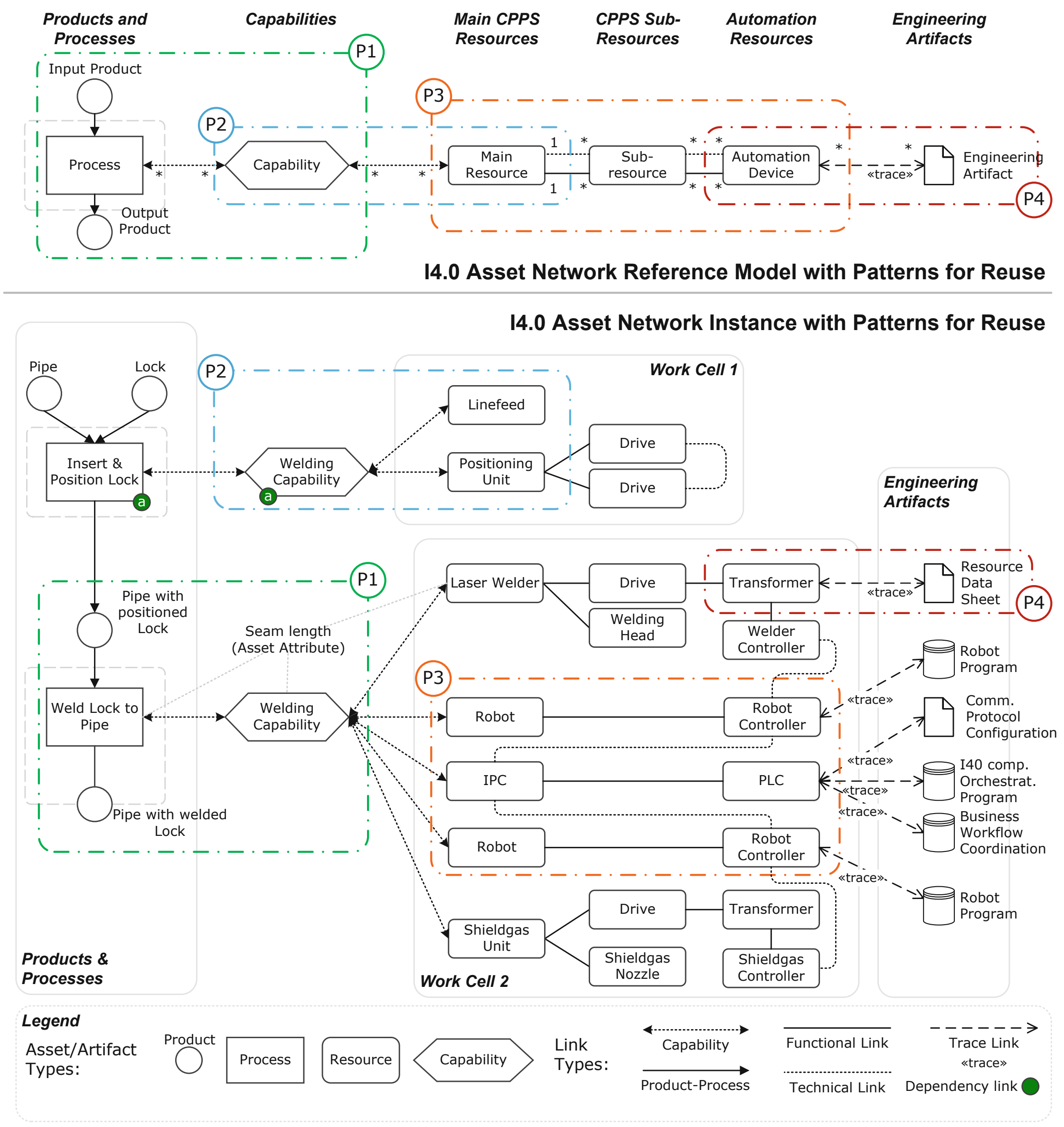


Figure 3.7: I4AN reference model (top row) and Patterns for Reuse (dashed contours) in an I4AN for the *shift fork* use case (cf. Section 1.1.1), in adapted VDI 3682 notation [199].

the delineation of concepts between multiple disciplines. The reference model is based on a typical CPPS engineering process and its engineering artifacts at an industry partner (cf. Section 1.1.1 and Section 1.1.2). Furthermore, it incorporates insights from a domain analysis in the automotive industry [72].

Figure 3.7 visualizes the I4AN reference model at the top of the figure and an instance of an I4AN for the *shift fork* use case (cf. Section 1.1.1) at the bottom. Furthermore, it illustrates four reusable engineering design patterns in dashed contours (*P1-P4*). The reference model defines the elements required to represent typical solution designs. The notation follows the extended VDI 3695 notation for PPR models with the following additions. Relations to and from Capabilities, i.e., from Processes and Resources, are shown as arrows with a dotted line. Functional Links are shown with a solid line, Technical Links with a dotted line, and Trace Links with a dashed arrow. Dependency Links are illustrated as small circles with a corresponding circle with the same label. Attributes of class NamedObject are shown near the particular entities connected with a dotted line.

The I4AN reference model specifies elements to identify particular engineering design patterns in engineering organizations. Therefore, the reference model focuses on a production process executed by one or more production resources. Figure 3.7 shows these elements in "columns" labeled with their membership to particular asset or artifact types, which we describe from left to right. We explain the reference model using the instance below.

The leftmost column shows input and output *Products* linked to a production *Process* that engineers need to define. For example, in the I4AN instance, the product *Lock* is inserted into the production line and positioned onto the *Pipe* in the *Insert & Position Lock* process. A Process may require several Capabilities that abstractly describe the requirements of the *Products* to be manufactured in that *Process*. For instance, welding the lock onto the pipe requires a *Welding Capability* with specific properties such as the welding *Seam length*. On the other hand, Resources provide particular Capabilities that may match. In this way, Capabilities represent interfaces between *Processes* and Resources, decoupling them in favor of more flexibility similar to software engineering. In the I4AN instance, *Work Cell 2* with its resources provides such a *Welding Capability*. Often, resources are structured hierarchically *Main Resources*, *Sub-resources*, and *Automation Devices* linked through Functional Links or Technical Links. For example, there is a *Main Resource Laser Welder*, connected to a *Sub-resource Welding Head*, and an *Automation Device Welder Controller*. *Capabilities* are typically related to the *Main Resources* in the CPPS. *Automation Devices* are related to *Sub-resources* or, sometimes, *Main Resources* to control their behavior. Finally, there exist *Engineering Artifacts* with data or information relevant to various assets that are connected to these assets via Trace Links. The *Robot Controller* requires a *Robot Program* for particular instructions on how to execute the process of *Welding*.

The presented elements and categories of the reference model were found relevant in the domain analysis and built the foundation for identifying engineering design patterns. From this, we identified four basic engineering patterns (cf. Figure 3.7 *P1-P4*) that engineers can use to elicit concrete engineering patterns for their domain and engineering organization. We described a simple procedure for each pattern to collect the relevant elements from the I4AN. With this engineers can create efficiently from a set of engineering artifacts that contain the required model information. Here, we give a brief overview of these engineering patterns.

**P1**. Product-Process-Capability: The engineers tasked with functional CPPS planning aim at selecting suitable and proven production processes with capabilities for (slightly) abstracted products. Therefore, we identified the *Product-Process-Capability* pattern as a self-contained entity as the foundation to identify concrete patterns of this kind. For instance, when plastic parts need to be solidly joined onto a metal part, the plastic only sustains a certain force and temperature. Suitable processes may include gluing or welding at low temperatures, which can be described as capabilities with stiffness and temperature constraints. Alternatively, a more specific capability, such as ultrasonic welding, can be defined for solid connections at low temperatures, creating a *low-temperature plastic joining Product-Process-Capability* pattern.

**P2**. Capability-Resource Pattern: Engineers aim to design resources that provide the capabilities potentially required to execute production processes based on their functions. To this end, we identified the *Capability-Resource* pattern as a reusable description of resource functionality. For example, a particular line feed can carry workpieces of a certain size at a certain velocity and accuracy from a box to an actuator like a robot. This linefeed can then provide, for instance, a transport capability, resulting in a potential *transport with velocity X Capability-Resource* pattern, where a type of linefeed with a provided transport capability can be matched to a required capability by a process.

**P3**. Resource-Resource Composition Pattern: Engineers in the design and implementation phase, i.e., detailed engineering, aim to plan, configure, and program production resources. This requires the reuse of blueprints of connected resource components. The *Resource-Resource Composition* pattern represents the composition of a resource from main to sub-resources and automation devices and their relations, incorporating technical parameters and dependencies. For instance, Figure 3.7 shows *Work Cell 2*, which has two robots that each require a robot controller that is driven by an industrial PC (IPC). With the *Resource-Resource Composition* basic pattern, engineers can identify the main resource and the strictly required sub and automation resources to reuse as a component that, nevertheless, is relatively independent of the other components.

**P4**. Resource-Artifact Pattern: To build similar types of CPPSs efficiently, engineers must reuse and configure as many engineering artifacts as possible. Furthermore, engineers shall be able to retrieve the relevant engineering artifacts for particular resources easily. The *Resource-Artifact* pattern connects crucial engineering artifacts to production resources, facilitating efficient reuse of resources and their associated data or programs. For example, a transformer might require an electronic data sheet where essential properties and their value range are denoted. Similarly, a positioning unit that positions larger workpieces always needs an acceleration curve artifact to stay in the safe acceleration range.

From the I4AN reference model and the basic patterns, engineers are encouraged to identify concrete reusable patterns in their organization in several ways in the DAE lifecycle. In the analysis activity, the engineers can use the basic patterns to analyze already existing CPPS artifacts and designs utilizing an extractive approach. In the design and implementation activities, the engineers can use the I4AN reference model and basic patterns to develop concrete engineering design patterns. Furthermore, they can elicit patterns and artifacts in a project retrospective. This way, the engineers can build up a common artifact repository. Beyond that, they can gradually build an engineering knowledge graph for CPPS engineering knowledge reuse.

Therefore, we implemented a prototypical application with Neo4J[10] and Neo4J Bloom[11], based on the I4AN knowledge model. Furthermore, we conducted a feasibility study with an industrial use case from the automotive industry originating from a domain analysis with 80 types of robot cells and 27 robot types [72]. We showed that engineering design patterns can be instantiated with the I4AN instance using boundaries and presented examples of applied patterns for reuse. Furthermore, we investigated queries to the I4AN and whether the queries retrieve valid patterns and discussed the results with industry partners.

*To summarize*, based on a domain analysis at a partner from the automotive industry, we built the I4AN reference model. The reference model can help structure CPPS engineering knowledge along the engineering lifecycle. This way, it aids in identifying reusable engineering design patterns, which serve as a basis for multidisciplinary knowledge coordination and communication. We identified four basic but essential engineering patterns – product-process-capabilities, capabilities-resources, resources-sub-resources-automation devices, and resources-engineering artifacts. Additionally, we implemented the I4AN reference model as an engineering graph in a graph database based on the I4AN knowledge model. Additionally, we developed several concrete engineering patterns and incorporated them into the engineering graph. This way, we build a common artifact repository with reusable engineering knowledge.

**Research Outcome.** The I4AN knowledge and I4AN reference model with the engineering design patterns provided the foundation for (i) further research on utilizing it as a coordinating artifact for multidisciplinary reuse of CPPSs engineering knowledge [134], (ii) several publications inside the research group [13, 16, 14, 168] and in research collaborations [112, 76], and (iii) applications in industrial research and development in two research projects regarding production quality improvement [103, 102] in an Austrian company and advanced data analytics [77] in a German company.

### 3.2.2 CPPS Design Reuse with C&Ss

The VDI 3695 procedure model [200] proposed, based on the DAE framework [152], a systematic approach for developing reusable production systems engineering artifacts in one and their instantiation and parameterzation in a second lifecycle. Meixner et al.

[10]Neo4J: `https://neo4j.com`
[11]Neo4J Bloom: `https://neo4j.com/product/bloom/`

[135] proposed the conceptual CSR framework that describes how the VDI 3695 needs to be adapted to CPPS engineering with C&Ss [149, 93, 58]. This framework aims to enable engineers to reuse engineering models and artifacts more systematically, exploiting C&Ss to decouple production processes and resources.

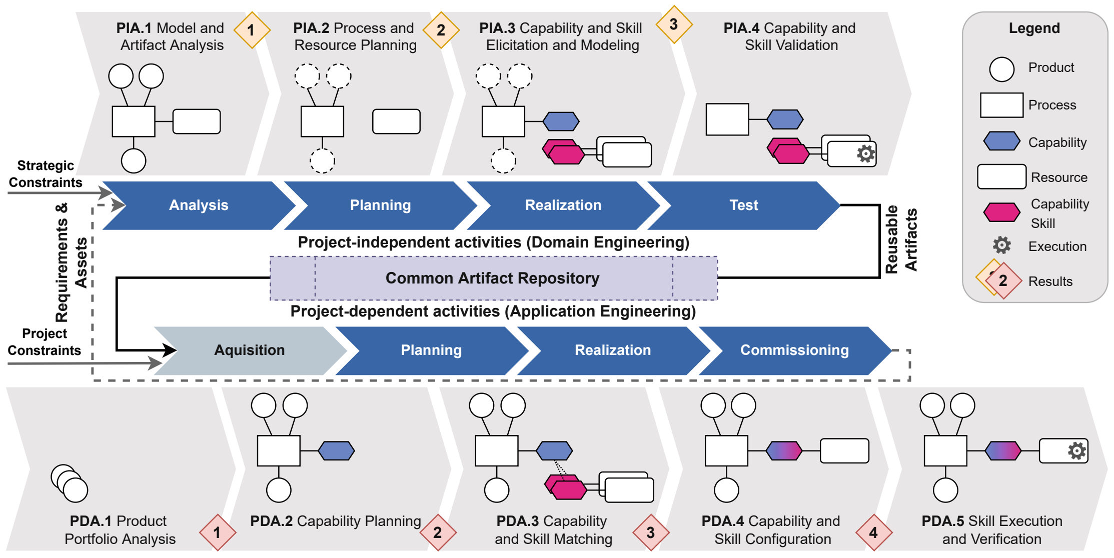

PIA.1 Model and Artifact Analysis
1
PIA.2 Process and Resource Planning
2
PIA.3 Capability and Skill Elicitation and Modeling
3
PIA.4 Capability and Skill Validation
Legend
Product
Process
Capability
Resource
Capability Skill
Execution
2
Results
Strategic Constraints
Requirements & Assets
Analysis
Planning
Realization
Test
Reusable Artifacts
Project-independent activities (Domain Engineering)
Common Artifact Repository
Project-dependent activities (Application Engineering)
Project Constraints
Aquisition
Planning
Realization
Commissioning
PDA.1 Product Portfolio Analysis
1
PDA.2 Capability Planning
2
PDA.3 Capability and Skill Matching
3
PDA.4 Capability and Skill Configuration
4
PDA.5 Skill Execution and Verification

Figure 3.8: The CSR framework for PPR models and artifacts with C&Ss, based on DAE [152] and the VDI 3695 [200, 83], from Meixner et al. [135].

Figure 3.8 shows the CSR framework with the typical VDI 3695 activities (in blue) and activities for reuse with capabilities and skills with PPR (in light gray). We enumerated the activities from domain engineering as PIA 1 to 4 and application engineering as PDA 1 to 5, responding to the VDI 3695 [200]. The following paragraphs discuss these activities and how they shall be conducted, considering the novel framework and carrying over the results from the previous activity to the next activity.

**PIA1**. *Model and Artifact Analysis.* In this activity, engineers assess the strategic constraints of their manufacturing domain. For instance, our industry partner holds a unique selling proposition in high-speed automation of CPPSs. Then, engineers have to analyze past engineering projects for reusable PPR engineering artifact candidates. In contrast to the traditional approach, engineers must identify (i) processes that can be abstracted from the products and (ii) resources whose functionality can be described as required, respectively, provided capabilities. This activity results in a domain reference, for instance, high-speed work line automation, and documented (partial) artifact candidates that contain processes and resources.

**PIA2**. *Process and Resource Planning.* Engineers take up the identified artifact candidates and separate the PPR assets in them into processes and resources. Furthermore, they categorize them following, for instance, domain-specific guidelines, such as the

DIN 8580 for processes.[12] The engineers generalize these PPR assets, assess their variability, and plan their configuration options and parameterization. The results are separated and categorized processes (with generalized products, such as two aluminum parts), resources, and capability candidates. In contrast, the traditional approach results in less modular and decoupled processes and resources.

**PIA3**. *PPR and C&S Modeling.* Engineers aim to realize the identified PPR assets as loosely coupled, reusable PPR and C&S models. The engineers represent a process asset as a process with a required capability and a resource asset as a resource with a provided capability. Additionally, the provided capability for a particular resource is implemented as a skill. Developing C&Ss requires appropriate models and languages, such as ontologies [96, 204] or DSLs [130] for capabilities and OPC UA [95, 215] or PackML[13] for skill implementation. The realized reusable and parameterizable models for PPR with C&Ss are deployed into the common artifact repository using an agreed-on repository structure. In the traditional approach, the engineers directly model the processes and, from them, the resources, resulting in increased coupling, which makes reconfigurability harder.

**PIA4**. *Model Validation.* The reusable artifacts must be instantiated and thoroughly tested and validated, for instance, on industrial testbeds or training plants. This activity is an important step towards a "qualification" for the rigorous (real-time) demands of CPPSs.

In a particular engineering project, the activities of application engineering, here named PDAs aligned with the VDI 3695, depicted in the lower part of Figure 3.8, are conducted.

**PDA1**. *Product Portfolio Analysis.* In this activity, engineers analyze the family of products intended to be manufactured on the CPPS. Therefore, the compare the product parts, for instance, the pipe and lock, to the abstracted product parts, for instance, the aluminum parts, and process artifact candidates in the common artifact repository that fit. This task results in a bill of products with the product family's corresponding variability and configuration. Furthermore, it results in a set of requirements for processes.

**PDA2**. *Process and Capability Planning.* Engineers investigate how to manufacture the products deriving process models. In contrast to the traditional approach, where resources are directly assigned, engineers search the common artifact repository for suitable reusable process models with their capabilities. Next, the engineers configure those capability models with the values of the products, such as the welding heat and speed rate. This task results in models of partially configured product, process, and required capability models.

**PDA3**. *Capability and Skill Matching.* Engineers match the partially configured process capability models with provided capabilities from the common artifact repository

[12]DIN 8580 – `https://standards.globalspec.com/std/1742169/DIN%208580`
[13]PackML – `https://www.omac.org/packml`

with computational support, resulting in a list of matches. Ideally, engineers can also select from various skills implemented for particular resources.

**PDA4**. *Capability and Skill Configuration.* Engineers iteratively configure the capability models for a final list of skills and resources. This step results in configured models and artifacts for a CPPS design. Furthermore, concrete artifacts, such as control code, can be generated from additional engineering artifact templates. In the traditional approach, processes are already bound to resources.

**PDA5**. *Skill Verification and Execution.* During verification and execution, the collected configured models and artifacts are tested. This can be done in a simulation or digital twin or if parts of the CPPS are already in the state of building on the system.

To summarize, the CSR framework refines DAE activities towards systematically using PPR and C&S models for decoupling products and processes from resources for improved reuse by (i) eliciting assets from prior projects abstracting them for reuse, (ii) representing them as PPR and C&S models and validate them as reusable artifacts in a common artifact repository, and (iii) using, configuring, and testing them in particular engineering projects [135]. Experiences, assets, and engineering artifacts, such as the PPR and C&S models, from the PDAs are fed back to the PIAs to improve the reuse lifecycle [135]. We compared the approach to the traditional reuse approach in these engineering organizations.

**Research Outcome.** We expect our industry partner to timely adopt the approach in their new engineering information system for transitioning from the traditional approach of CPPS engineering toward DAE activities. In research, concepts of the approach were included in the effort to build a common community model for C&Ss [97] in the CPPS engineering community.

## 3.3 Integrated Reuse and Variability Management for CPPS Engineering

In the previous section we described contributions that allow engineers to systematically communicate externalized knowledge. Furthermore, we presented an approach for reusing engineering knowledge along the DAE cycles.

This section presents three essential contributions to further advance an integrated approach for reuse and variability management for CPPS engineering. The first section presents the VERT approach enabling CPPS engineers to transform engineering artifacts with variability to state-of-the-art variability models. For instance, we transform a TCM a feature model, which engineers can change and transform back to the TCM via VERT for iterative development. We then present four reusable real-world use cases of different complexity modeled in the PPR–DSL and transformed to the de-facto standard feature models. We use the TRAVART variability transformation approach as an open source baseline to compare variability models. Finally, we describe the IPSE and EIPSE approaches that provide an integrated reuse and variability management approach for CPPS families using three interdependent families of Product-Process-Resource.

### 3.3.1 PPR Variability Modeling and Transformation

**Basic Variability Modeling and Engineering Artifact Transformation.** While there are several SPL methods and techniques, including variability modeling, they are not yet well adopted in industry and, especially, in production systems engineering [6]. For modeling variability, stakeholders from the industry often develop custom-tailored variability models and artifacts that suit their purposes. A typical artifact for representing the variability of products are TCMs (cf. Table 1.1), which lack semantics and are hard to validate.

Therefore, we designed the VERT approach [43] for iterative round-trip transformation. This approach enables CPPS engineers to transform their custom-developed TCMs to a state-of-the-art feature model. This way, we aim to counter the lack of semantics and validation issues but also scattered and implicit knowledge. The approach utilizes TRAVART, a recent transformation approach [42], which aims for a pivotal transformation between well-established and custom variability models. VERT also allows the definition of preconditions for assembling products in a precedence graph to create the additional constraints for the feature model. VERT supports round-trip transformation to evolve and optimize the feature model and transform it back to its original format, i.e., the TCM. We evaluated the approach iteratively with the *shift fork* use case (cf. Section 1.1.1). This activity resulted in several versions of the shift fork feature model comparable to the feature model in Figure 3.13a. Furthermore, we conducted a workshop with domain experts from our industry partner. With them, we derived lessons learned and potential improvements toward an integrated reuse and variability management approach.

The main findings were that (i) the engineers recognized that production processes and, in particular, their sequence should be represented in structured variability models, (ii) an approach to evolve the engineering artifacts and variability models must be iterative, (iii) the product type information needs to be maintained in the variability models, and (iv) tool support is essential for engineering artifact transformation and managing variability as the foundation for reuse. In this regard, we gained valuable insights and feedback for the research endeavor. Furthermore, we identified additional challenges of the variability artifact transformation. These challenges include precise mappings between the custom variability and the feature model to maintain relevant CPPS engineering information like identifiable products.

*To summarize*, although many SPL methods and models exist, their adoption in CPPS engineering is limited. Therefore, industry stakeholders often create custom variability artifacts, like TCMs, that lack semantics and are hard to validate. With the VERT approach we address these issues by enabling iterative transformations of TCMs into feature models using TRAVART. This way, we improve semantics and validation while supporting round-trip transformations. We evaluated the approach with the *shift fork* use case and in a workshop with the main engineers. The findings showed a positive recognition of the approach but also revealed challenges, such as the need to preserve essential CPPS information in future models.

**Advanced PPR Variability Modeling and CPPS Case Studies.** Based on the gaps identified, i.a., in the VERT publication [43], we have established the PPR–DSL [130, 123] for more advanced PPR and variability modeling (cf. Section 3.1.2). Furthermore, we aimed to enhance our evaluations and carry out further empirical studies of our approaches within industrial case studies to ensure comparability. Beyond that, we aimed to provide the case studies' data to other researchers and practitioners, which are otherwise often not accessible [115]. This dissemination follows our commitment to open science and reproducibility, especially for models of industrial product families.

To address these issues we employed four reusable real-world case studies for product family in production systems with different complexity in Meixner et al. [127]. The case studies are the *truck*, *shift fork*, *water filter*, and *rocker switch* case studies.

- The **truck** use case comprises a 3D-printed truck product family. The product family is manufactured on the academic *Testbed for Industry 4.0 (I4.0)* at Czech Technical University in Prague with three robot arms and a conveyor belt that can be flexibly configured [127].[14] The *truck* use case is of low complexity, with just four possible types and no dependencies between the features in the product structure itself.
- The **shift fork** product family is a use case from one of our industry partners explained in detail in Section 1.1.1 [127]. The *shift fork* use case is of low complexity, consisting of four different shift fork types with three constraints among the products and manufactured on a timed conveyor-belt CPPS.
- The **water filter** use case covers the low-cost, locally manufactured, and configurable *NanoFilter water filter* product family. It originates in a frugal Tanzanian development project, where researchers investigated different filter materials and constructions [94].[15] This way, the electricity-free water filter filters impurities and can selectively remove contaminants from unsafe water sources, addressing the basic needs of price-sensitive customers in developing countries [127]. We consider the use case with its eight different product types and around 160 dependencies of medium complexity.
- The **rocker switch** product family is a use case from one of our industry partners explained in detail in Section 1.1.1. We consider the *rocker switch* use case, manufactured on a timed conveyor-belt CPPS, of medium complexity consisting of twelve different types with around 180 dependencies among the products [127].

[14]Industry 4.0 Testbed: `https://ciirc.cvut.cz/teams-labs/testbed/`
[15]Gongali Project: `https://gongalimodel.com/`

With the support of the particular case's domain experts, we refined the case studies toward variability modeling and reuse of the products. Therefore, we modeled the case studies in several iterations in the PPR–DSL. This activity resulted in artifacts with variability and precise semantics, making them uniformly accessible and human-readable. Table 3.1 shows the statistics of these use cases along with their categorized complexity. Furthermore, it includes the number of product types, the number of PPR–DSL elements required to model the product family, and the number of dependencies in the PPR–DSL model.

| **Use Case** | **Complexity** | **#Product Types** | **#DSL Elements** | **#Dependencies** |
|---|---|---|---|---|
| 3D-printed truck | low | 4 | 12 | 31 |
| Shift fork | low | 4 | 22 | 36 |
| Water filter | medium | 8 | 54 | 165 |
| Rocker switch | medium | 12 | 54 | 184 |

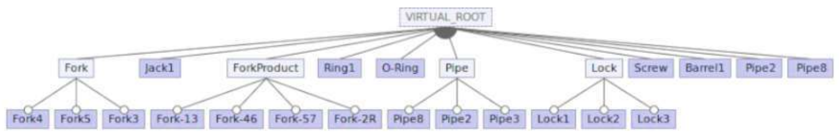
Table 3.1: Overview of the characteristics of the four case studies.

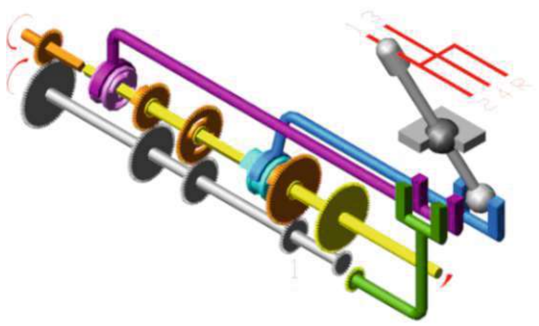
These PPR–DSL models provided the foundation to derive product variability models using TRAVART [46]. However, this direction also required additional model transformations for an automated transformation from the PPR–DSL to the variability models, i.e., feature models. Based on learnings from the previous publication [43], we mapped the product elements and their variability concepts from the PPR knowledge model to FeatureIDE [116] feature models. A model mapping for the transformation of the PPR–DSL to feature models that are more advanced is shown in Figure 3.12. However, the model mapping for this publication is left out here for brevity but can be found in the publication [43].

We subsequently implemented the TRAVART transformation operations [46] to the FeatureIDE feature model's XML syntax and conducted the transformations on the four case studies. Figure 3.9 shows, as an example, the renderings of the shift forks (in blue, magenta, and green) in a manual transmission on the bottom left, a section of the shifts for products and their parts on the bottom right in the PPR–DSL, and the resulting feature model on top of the figure. We published the case studies' data and their artifacts in a Git repository.[16] Furthermore, we compared and evaluated the completeness and expressiveness of the PPR models and the feature models, ensuring they were transformed without information loss.

However, we also identified the following limitations of the approach, as the feature model at the top of Figure 3.9 shows. First, the feature model maintains the final product

[16]GitHub Repository: `https://github.com/tuw-qse/cpps-var-case-studies`





```
Product "Fork-13": {
    name: "Fork 13",
    implements: ["ForkProduct"],
    requires: ["Pipe8", "Lock3"]
}

Product "Fork-2R": {
    name: "Fork 2R",
    implements: ["ForkProduct"],
    requires: ["Pipe3", "Lock2"]
}

Product "Fork-46": {
    name: "Fork 46",
    implements: ["ForkProduct"],
    requires: ["Pipe3", "Lock1"]
}

Product "Fork-57": {
    name: "Fork 57",
    implements: ["ForkProduct"],
    requires: ["Pipe2", "Lock1"]
}
```

Figure 3.9: Rendered *shift forks* (in blue, magenta, and green) in a manual transmission, (Image: World Arm Lamp, CC0 on Wikipedia - `https://w.wiki/3DCf`), a section of the *shift fork* products in PPR–DSL, and the resulting feature model, from Meixner et al. [127].

types required for round-trip engineering. However, instead of "not showing" them in the configuration, the final product types can be selected as features in the configuration. Second, while there are constraints concerning which features exclude and imply each other, the features miss *or* and *xor* groups and are defined as *optional* features missing the *mandatory* relations in the feature groups. Finally, the case studies only describe the product variability rather than the three main families of CPPSs and do not address the variability in processes and resources.

*To summarize*, we described four real-world case studies, the 3D-printed *truck*, *shift fork*, *water filter*, and *rocker switch* case studies. We aimed to use these case studies as a baseline for empirical evaluations and reproducibility. We addressed the identified gaps in the TCMs of unclear semantics, limited verification, and scattered engineering artifacts. Therefore, we modeled the case studies in our previously developed PPR–DSL [130, 123] with their variability, making them accessible and human-readable. These models enabled the derivation of detailed product feature models with TRAVART, highlighting the diverse configurations within each product family. Additionally, we transformed the models back to the PPR–DSL and validated them for completeness and expressiveness [43]. This way, we achieved a round-trip transformation including the initial products, as identified in a requirement in a prior publication [43].

**Research Outcome.** The results of the variability transformation and the use cases were picked up (i) by our subsequent research efforts [123, 130, 127, 132, 137, 138, 64],

(ii) as part of a PhD thesis [41] showcasing the successful transformation of custom variability artifacts from industry to well-established variability models, and (iii) partially by the TRAVART [46] transformation approach.[17] Beyond our collaboration, this work and subsequent work [138] led to discussions in the SPL community on how to extend the recent community-effort variability language UVL [192]. For dissemination, we made the case study artifacts publicly available in a Git repository to support open science and reproducibility.[18] The four use case studies were also published in the ESPLA catalog [115], a collaborative catalog of case studies for software product line adoption and extraction.[5]

### 3.3.2 Integrated Reuse and Variability Management

The IRVM research line (**RL3.**) aimed to achieve the following objectives. Enable modeling and managing the three essential interdependent product families of products, processes, and resources for engineering CPPS families. In particular, this should include the *configurable and flexible* sequence of production steps [85, 143], which reflects the behavioral variability of the CPPSs. Overcome the manual modeling and configuration process, based on *implicit domain knowledge* that comes primarily *from experience and undocumented dependencies*, which is *hard and, most of the time, impossible to reproduce* [132, 137]. Maintain the paramount *separation the concerns* in the engineering process including variability modeling and reuse management [1, 132, 137].

We addressed these challenges, building on our PPR knowledge model for CPPS engineering the PPR–DSL (cf. Section 3.1 and our variability modeling and transformation approaches (cf. Section 3.3.1). Therefore, we developed the IPSE in Meixner et al. [132] and extended it in the EIPSE approach in Meixner et al. [137].[19] Figure 3.10 shows the overall approach with steps from EIPSE.[20] Furthermore, we developed the EIPSE toolchain architecture for the approach, which Figure 3.11 illustrates, and implemented it as the EIPSE tool.[21] In the following, we describe the EIPSE process along with a description of the components in the toolchain.

The EIPSE process starts, similar to the traditional engineering process, with receiving the product specifications and descriptions. From them, in Step 1 of the EIPSE process, CPPS engineers iteratively model a PPR–DSL model (cf. Listing 1). Ideally, the engineers select and slightly adapt atomic process steps and production resources from a common artifact repository, as explained in Section 3.2.2. In the toolchain architecture, the PPR–DSL component is shown in yellow at the top left.

[17]TRAVART Github: `https://github.com/SECPS/TraVarT`

[18]Variability Case Studies: `https://github.com/tuw-qse/cpps-var-case-studies`

[19]Additional material to EIPSE can be found under: `https://github.com/tuw-qse/eipse`

[20]Steps from IPSE are in dashed contours, updated steps with solid contours, and novel steps with solid contours and a darker color.

[21]The color coding follows the same terminology of the EIPSE process. Existing components are displayed in dashed contours, updated ones with solid contours, novel ones with solid contours, and additional components in a darker color.

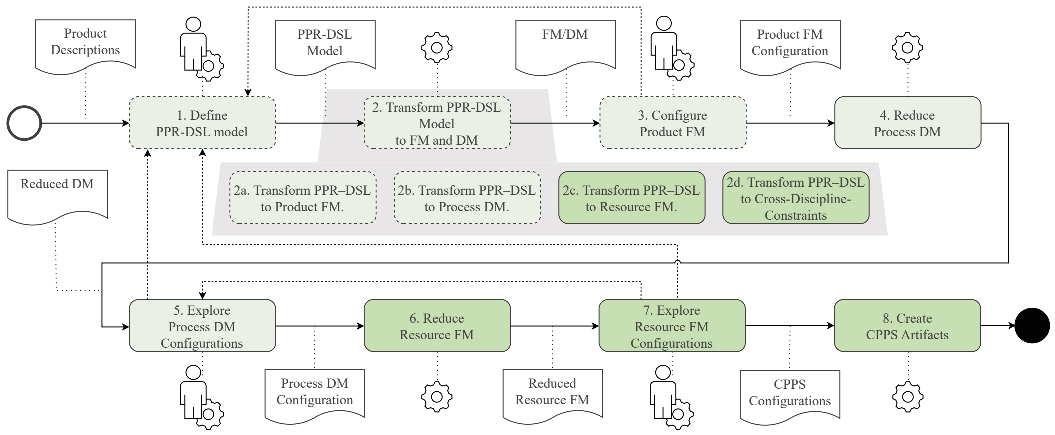

Product Descriptions
PPR-DSL Model
FM/DM
Product FM Configuration
1. Define PPR-DSL model
2. Transform PPR-DSL Model to FM and DM
3. Configure Product FM
4. Reduce Process DM
Reduced DM
2a. Transform PPR–DSL to Product FM.
2b. Transform PPR–DSL to Process DM.
2c. Transform PPR–DSL to Resource FM.
2d. Transform PPR–DSL to Cross-Discipline-Constraints
5. Explore Process DM Configurations
6. Reduce Resource FM
7. Explore Resource FM Configurations
8. Create CPPS Artifacts
Process DM Configuration
Reduced Resource FM
CPPS Configurations

Figure 3.10: (Human & automated) EIPSE process steps [137] for exploring production process steps based on a product configuration (steps from IPSE [132] in dashed contours, updated steps with solid contours, and novel steps, additionally in a darker color), from Meixner et al. [137].

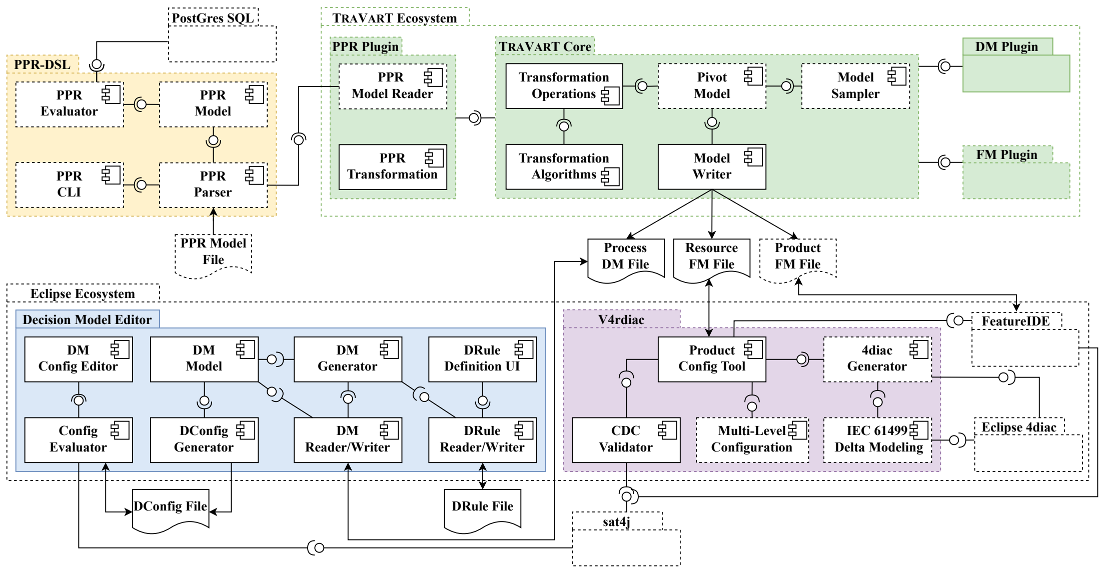


Figure 3.11: Architecture of the EIPSE toolchain in UML component diagram notation (novel and updated components with solid contours), from Meixner et al. [137].

In Step 2, EIPSE uses TRAVART [46] to automatically transform the PPR–DSL model into suitable variability models. Modeling the interdependent product families meant linking their structural and behavioral variability utilizing well-established variability models and models to capture their dependencies.

Therefore, in the IPSE [132] we transformed the PPR–DSL into (i) a Product FM

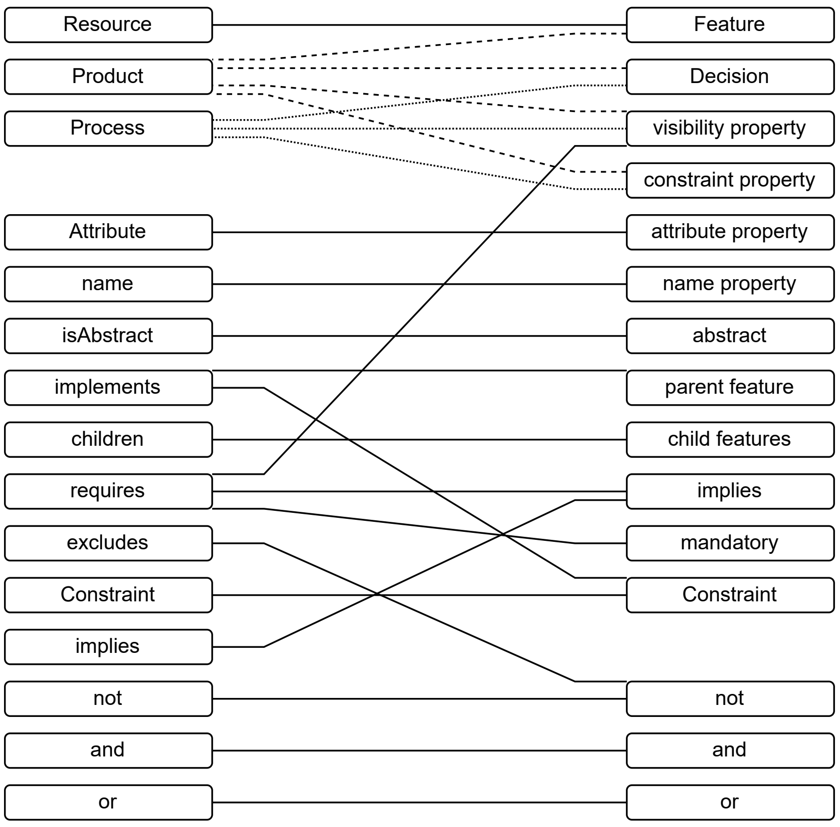


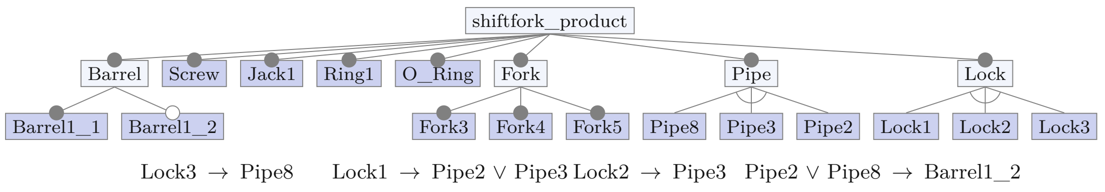
Figure 3.12: Conceptual mapping of the PPR–DSL knowledge model to UVL [192] for creating feature and decision models.

to represent the product types, their parts, and structural variability (Step 1a) and (ii) a DOPLER Process DM [68, 30] to represent the atomic process steps and their dependencies for aThe partial behavior of the CPPS (Step 1b), and (iii) automatically linked both models (Step 4) [132]. The EIPSE approach additionally transforms the PPR–DSL into (i) a Resource FM to represent the hierarchical structure of production resources (Step 2c), and (ii) CDCs capturing the dependencies between the three models in propositional logic (Step 2d).

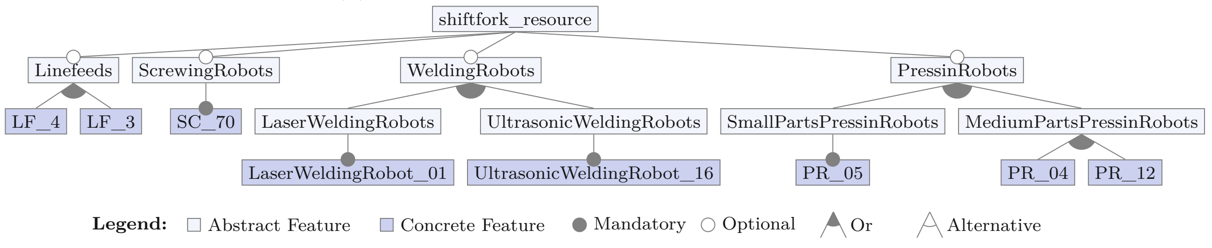
Figure 3.12 shows the model mapping from the PPR–DSL to UVL [192] creating FeatureIDE feature models and DOPLER decision models. We briefly describe this mapping in the following, beginning with the high-level concepts and advancing to their properties. *Resources* are mapped to *features* in the UVL Resource FM. Similarly, *Products* map to *features* in the UVL Product FM. Furthermore, they are mapped to *decisions* that are not visible to configurators due to their *visibility properties* in the UVL Process DM. The dependencies between the products are modeled as *constraint rules*. These additional properties and constraints are required for the satisfiability calculation of the model. *Processes* are mapped to decisions. *Attributes* of the PPR–

DSL model are automatically transformed to general *attribute properties* in the UVL. The *IDs* of resources, products, and processes are transformed to *feature names*, and the names of resources, products, and processes are transformed to name properties. The *isAbstract* attribute of the PPR concepts is transformed to *abstract* features and decisions. The *implements* list of the PPR concepts are translated to *parents* features, or if several concepts are used in the list to *constraints*, as feature and decision models do not allow several parents. The *children* list is transformed to *child features*. The requires list is transformed either to visibility conditions, *implies* constraints, and *mandatory* features necessary. The elements in the *excludes* list are transformed to *constraints* and concatenated with *not*. *Constraints* are straightforward transformed to UVL *constraints* with *not* mapping to *not*, *and* mapping to *and*, and *or* mapping to *or*.

In the toolchain architecture, we advanced the PPR transformations in the PPR plugin of TRAVART with the transformation operations, and the transformation algorithms. Furthermore, we changed the decision model plugin to better support visibility conditions and constraints in the DOPLER decision models. Additionally, we changed the model writer for the different variability model files. With the model mappings and corresponding TRAVART transformation operations, we transformed the PPR–DSL models to the respective variability models. Figure 3.13 shows the resulting Product FM for the shift fork parts at the top and Resource FM for potential shift fork resources at the bottom (cf. Section 1.1.1). Table 3.2 shows the corresponding Process DM for the shift fork.



(a) Product FM of the shift fork parts



(b) Resource FM of the production resources for the shiftfork CPPS

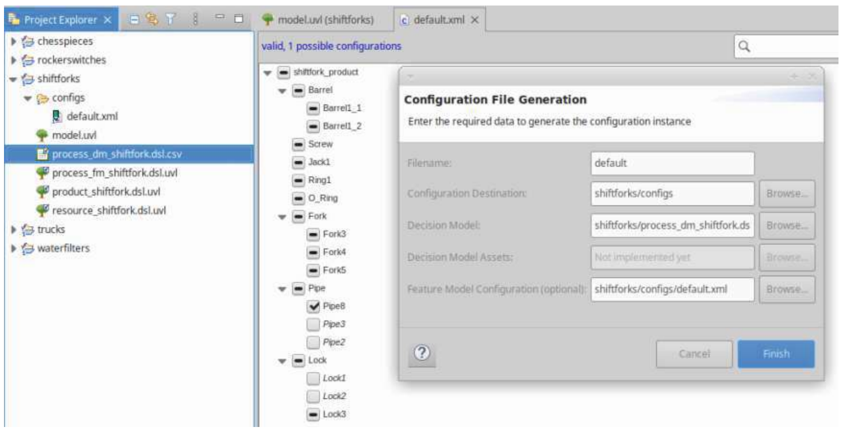
Figure 3.13: FeatureIDE feature models [116] of product (top) and production resource (bottom) variability of the *shift fork* use case.

In Step 3, a CPPS engineer configures the Product FM, which results in a valid product configuration. Engineers can conduct this task in the standard FeatureIDE configurator.

Step 4 automatically "reduces" the Process DM configuration based on the Product FM configuration to exclude decisions not needed for the configured product.

| ID | Question | Type | Range | Card. | Visible/Relevant if | Constraint/Rule |
|---|---|---|---|---|---|---|
| Pipe | Which Pipe types? | Enum | Pipe2 \| Pipe 3 \| Pipe8 | 1:1 | false | |
| Barrel1_2 | Install Barrel1_2? | Bool | true \| false | | false | |
| Lock | Which Lock types? | Enum | Lock1 \| Lock2 \| Lock3 | 1:1 | false | Lock1 $\Longrightarrow$ Pipe = Pipe2 $\vee$ Pipe = Pipe3 Lock2 $\Longrightarrow$ Pipe = Pipe3 Lock3 $\Longrightarrow$ Pipe = Pipe 8 |
| ... | ... | ... | ... | ... | ... | ... |
| InsertPipe | Install InsertPipe? | Bool | true \| false | | false | |
| InsertPipe2 | Install InsertPipe2? | Bool | true \| false | | Pipe == Pipe2 | InsertPipe2 $\Longrightarrow$ InsertPipe |
| ... | ... | ... | ... | ... | ... | ... |
| InsertLock | Install InsertLock? | Bool | true \| false | | false | |
| InsertLock1 | Install InsertLock1? | Bool | true \| false | | Lock == Lock1 | InsertLock1 $\Longrightarrow$ InsertLock |
| InsertLock2 | Install InsertLock2? | Bool | true \| false | | Lock == Lock2 | InsertLock2 $\Longrightarrow$ InsertLock |
| ... | ... | ... | ... | ... | ... | ... |
| InsertBarrel1_2 | Install InsertBarrel1_2? | Bool | true \| false | | Barrel1_2 | |
| PressBarrel1_2 | Install PressBarrel1_2? | Bool | true \| false | | Barrel1_2 & InsertBarrel1_2 & InsertPipe | |
| ... | ... | ... | ... | ... | ... | ... |

Table 3.2: Excerpt of the generated DOPLER DM [30] representing the process variability of the *shift fork* case study.



Configuration File Generation

Enter the required data to generate the configuration instance

| | |
|---|---|
| Filename: | default |
| Configuration Destination: | shiftforks/configs |
| Decision Model: | shiftforks/process_dm_shiftfork.ds |
| Decision Model Assets: | Not implemented yet |
| Feature Model Configuration (optional): | shiftforks/configs/default.xml |

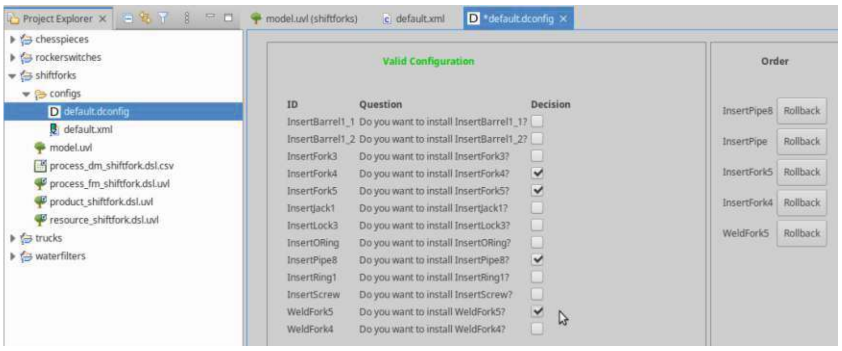
Figure 3.14: After configuring the Product FM in Step 3 of the EIPSE process (background) for the *shift fork* use case [130], the reduced Process DM configuration is created in Step 4 of the EIPSE process using the EIPSE prototype's wizard (front), from Meixner et al. [137].

In Step 5, a CPPS engineer can explore the remaining process decisions iteratively and interactively. However, the IPSE approach had *no automation and tool support to explore and configure process sequences* [137].

Therefore, we implemented the novel decision model editor (blue on the bottom left), embedded in the Eclipse ecosystem, to work seamlessly with the state-of-the-art variability modeling editor FeatureIDE. The editor not only allows the configuration of Process DM, but can also be used to create various kinds of decision models and their configuration. This means that, to the best of our knowledge, the editor is the first openly available editor for DOPLER decision models. With this, the EIPSE tool reduces the Process DM, which a CPPS engineer can then explore and configure based on a constant evaluation with SAT4J, the standard solver in the FeatureIDE.[22] As a result, the tool only displays process steps feasible in the particular configuration stage. In the background, the editor evaluates the visibility conditions and sets the subsequent configuration options based on the constraints in the Process DM. Figure 3.14 and Figure 3.15 show the decision model editor during the shift fork configuration.

In Step 6, the EIPSE tool automatically reduces the Resource FM configuration based on the previous Process DM configuration, resulting in a partial Resource FM configuration. In this configuration, possibilities are valid production resource configurations for a particular product and process sequence configuration from previous steps.

Based on this Resource FM, in Step 7, a CPPS engineer can configure the desired production resources with the EIPSE tool. However, the IPSE toolchain *did not include production resource modeling and configuration*. Therefore, we adapted and connected the closed source V4rdiac component of Fadhlillah et al. [39] (shown in violet on the bottom right), a co-author of the EIPSE publication, to exemplify the EIPSE approach. The V4rdiac tool takes up the CDC file, containing further dependencies between the different variability models, reduces the Resource FM, and enables its configuration.



| ID | Question | Decision |
|---|---|---|
| InsertBarrel1_1 | Do you want to install InsertBarrel1_1? | ☐ |
| InsertBarrel1_2 | Do you want to install InsertBarrel1_2? | ☐ |
| InsertFork3 | Do you want to install InsertFork3? | ☐ |
| InsertFork4 | Do you want to install InsertFork4? | ✓ |
| InsertFork5 | Do you want to install InsertFork5? | ✓ |
| InsertJack1 | Do you want to install InsertJack1? | ☐ |
| InsertLock3 | Do you want to install InsertLock3? | ☐ |
| InsertORing | Do you want to install InsertORing? | ☐ |
| InsertPipe8 | Do you want to install InsertPipe8? | ✓ |
| InsertRing1 | Do you want to install InsertRing1? | ☐ |
| InsertScrew | Do you want to install InsertScrew? | ☐ |
| WeldFork5 | Do you want to install WeldFork5? | ✓ |
| WeldFork4 | Do you want to install WeldFork4? | ☐ |

| Order | |
|---|---|
| InsertPipe8 | Rollback |
| InsertPipe | Rollback |
| InsertFork5 | Rollback |
| InsertFork4 | Rollback |
| WeldFork5 | Rollback |

Figure 3.15: During the configuration of the production steps in the Process DM in Step 5 of the EIPSE process, a suitable production process sequence for the *shift fork* use case [130] is defined. The EIPSE prototype provides a *rollback* option (right-hand side) for systematic process sequence exploration [133], from Meixner et al. [137].

Finally, in Step 8, we support generating particular CPPS engineering and operation artifacts. Based on which artifacts shall be generated, it requires an additional *artifact generation* component. For the EIPSE tool, we utilized V4rdiac to generate IEC 61499 control software [82]. V4rdiac picks up the configurations of the Process DM and Resource FM and parameterizes the base implementation of IEC 61499 code with the values from the configurations and generates the final control software. This software then contains the relevant processes and resources and can be further modified by CPPS engineers and then deployed to a manufacturing execution system and particular resources, such as robot arms.

[22]SAT4J: `http://sat4j.org/`

To evaluate the EIPSE approach, we conducted an empirical evaluation that involved: (i) *assessing the feasibility* of selecting a suitable CPPS variant in four previously presented real-world case studies [130], (ii) applying the EIPSE approach in the *novel* chess piece *case study conducted by domain experts* with diverse backgrounds and no prior experience with the EIPSE approach, and (iii) exploring the joint use of feature and decision models for configuring CPPS and *generating CPPS artifacts* with V4rdiac [39].

| Subject | | Activity | | | | | |
|---|---|---|---|---|---|---|---|
| | | Intro | Engineering | | Configuration | | |
| Participant | Profession | | Definition | Updates | Product FM | Procss DM | Resource FM |
| S1 | E,ME | 10m | 60m | 30m | 3m | 5m | 1m |
| S2 | E,ME | 13m | 70m | 15m | 2m | 7m | 3m |
| S3 | E,ME | 9m | 72m | 3m | 2m | 5m | 4m |
| S4 | E,SE | 6m | 62m | 4m | 1m | 4m | 1m |
| S5 | R,ME | 9m | 63m | 9m | 6m | 9m | 3m |
| Summary (avg) | | 9m | 65m | 12m | 3m | 6m | 2m |

Table 3.3: "Time spent" by engineers with different backgrounds for EIPSE to the *chess piece* case study. E. . . Engineer, R. . . Researcher; ME. . . Mechanical engineering, SE. . . Systems engineering, from Meixner et al. [137].

In the study, we assessed how much effort (senior) domain experts from diverse backgrounds and inexperienced with the EIPSE approach dedicated to each step of its application. Therefore, we utilized the *chess piece* case study, which we elicited from TU Wien's pilot factory,[23] where a CPPS produces the six types of chess pieces with different configurations. Five subjects participated in the study, modeled the chess pieces, the production processes, and the resources using the PPR–DSL, and configured the resulting variability models subsequently to the EIPSE approach. Table 3.3 shows the domains and origin of the subjects (mechanical (*ME*) or (*SE*) systems engineering; engineers (*E*) or researchers (*R*), how much time they spend on particular tasks, and a *summary* row that shows the "time spent" as average. We introduced the subjects to the EIPSE approach and the case study, spending an average of 9 minutes on this introduction. The subjects spent most of the time defining the *chess piece* product line, averaging 65 minutes. Additionally, on average, they spent 12 minutes updating their models according to feedback loops, for instance, when they found out that modeling a product as an abstract product makes sense. The configuration of the product, process, and resource variability models took around 3 minutes for the product and resource models. In contrast, the process sequence configuration took slightly longer, at 6 minutes on average.

[23]Pilotfabrik TU Wien: `https://www.pilotfabrik.at`

Our pre-modeled baseline for the *chest piece* case study comprises the 6 chess piece types with 17 features in the Product FM. Furthermore, it contains around 11 production process steps with 19 process decisions in the Process DM, and 5 production resources in the Resource FM. Following our categorization of the reusable use cases published before in Meixner et al. [127] (cf. Table 3.1), we categorize the complexity of this case study still *low*.

The users spent a large part of the time *defining* and *updating* the PPR–DSL models, compared to the configuration of the interdependent Product FM, Process DM, and Resource FM. We strongly assume that some time is owed due to the unfamiliar DSL. Considering this, these results show that the subjects required more time for the domain engineering activities than the configuration part of the application engineering activities. However, we argue that engineers, in any case, have to invest a significant amount of mental work to identify how to manufacture the product and which production processes and resources to use in the first place. Furthermore, with this externalization of their implicit knowledge into the PPR–DSL, they make their knowledge on the imagined CPPS available (i) for reusability, for instance, as libraries, increasing the reusable artifacts, (ii) a further iteration and evolution of the products and CPPS design lowering domain expert effort, and (iii) communication and transfer to other disciplines, as well as clients, for better reproducibility. This result indicates that the manual approach unlikely pays off for CPPSs families with low complexity. This concerns, for instance, the *truck*, *shift fork*, or *chess piece* use cases, except in projects where the products or the production system changes regularly. However, in medium to large projects with many dependencies, modeling, for instance, an additional product seems a marginal change in domain engineering while reusing the former models. Similarly, the user study indicates that the effort of additional exploration of the Product FM, Process DM, and Resource FM stays within reasonable limits.

This evaluation *strengthens our initial assumption* that an integrated approach for reuse and variability management allows more efficient engineering of CPPS families compared to the traditional approach (cf. Section 1.1.2). The approach does so by (i) *facilitating multidisciplinary knowledge exchange* via PPR model and engineering design patterns while separating the disciplines's concerns (C5., M2., M5.), (ii) *increasing reusable artifacts* that serve as a foundation for engineering iterations or future CPPSs (C1., M3.), and thus (iii) *reducing domain expert effort* when evolving or designing CPPS (C2.). Furthermore, through this approach, we expect an overall increase in the quality of engineering artifacts and the engineering lifecycle (C1., M2.).

*To summarize*, building on the previously established PPR knowledge model and PPR–DSL, we developed the IPSE and its extension, the EIPSE approach with the EIPSE toolchain. The EIPSE approach aimed to address, in particular, challenge subsumed in challenge **CH3**. aiming to improve the efficiency of reuse and variability management in CPPS engineering.

The EIPSE approach supports automated transformations from the PPR–DSL models to t interdependent variability models, i.e., the Product FM, the Process DM, and the

Resource FM. This way, the approach captures the structural and behavioral variability of CPPSs. Furthermore, EIPSE provides tool-assisted configuration tasks, including production process sequence and resource explorations based on a product configuration. Additionally, we showed the generation of parameterized CPPS artifacts with IEC 61499 control code [82] generation from the variability model configuration.

A user study with (senior) domain experts from academia and industry indicated that the EIPSE approach is effective. For a use case of low complexity, they spent more time in domain engineering, creating reusable PPR–DSL models. In contrast, the subsequent configuration of the three different variability models and the control code generation were relatively fast in the application engineering activity. The externalization of their engineering knowledge and separation into the different variability models in the study suggest that the approach facilitates multidisciplinary knowledge exchange. Furthermore, it increases the reasonability of engineering artifacts. We argue, that these findings show that the approach overall reduces the effort required from domain experts when evolving or designing CPPSs and their families.

**Research Outcome.** This work advances the state of the art by introducing a systematic reuse and variability management approach utilizing CPPS engineering models. We go beyond the state of the art as the utilized decision models significantly exceed the size and complexity of decision models commonly found in the literature [182]. In this regard, we can present a new baseline for decision models for the SPL community. For dissemination, we created additional material[19] along with a video of the IRVM approach in action.[24]

[24]eIPSE demonstration video: `https://youtu.be/eoNNDOusXKA`

CHAPTER 4

# Conclusion

This chapter discusses the research results and concludes the thesis.

## 4.1 Discussion

This section analyzes our research findings. To this end, it discusses the implications of the developed methods and models. Furthermore, it addresses the broader impact of the adoption of integrated reuse and variability management approaches in the industry.

Engineering organizations in CPPS engineering face, beyond others, the rising demand for more flexible and adaptable CPPSs. This flexibility subsequently requires reusing engineering artifacts and modeling the variability within these systems. This, in particular, concerns the different dimensions of variability, i.e., product, production process, and production resource variability.

**Variability Awareness.** Our industry partners are aware of the variability within a single CPPS but also in related CPPS. Concerning this variability, they face similar issues to other industrial engineering organizations [6]. According to an interview with a senior researcher in automation engineering, beyond others, these issues concern the multidisciplinarity of the field with implicit and scattered knowledge, variability on multiple levels, and long-running project times on long-living ecosystems.

As a result, engineering organizations use, for instance, TCMs to model the variability of the products. Furthermore, they use technical components from libraries to represent production resource variants. However, these libraries must be filled and maintained in parallel with their project business. There is also limited awareness of the holistic and interlinked variability of products, processes, and resources. This limited awareness is because thinking in multidisciplinary engineering objects over disciplined borders is the foundation for interdisciplinary and secure reuse. However, this relevant knowledge has only been acquired in recent years [181, 150]. Additionally, there is limited knowledge on SPL methods, which is, if known, seen as an opportunity but also as a significant challenge to implement, among other things, because of missing integrated tool chains.

To this end, the PPR model has already brought more insight into our industry partners. The superimposed PPR model for the visual representation of variability, presented in Section 3.1, brought additional awareness of the interrelated CPPS variability and its modeling to the engineers, helping them to communicate their engineering knowledge better. However, according to our industry partner, the rising complexity of the superimposed PPR model requires additional tool support and a pragmatic approach to represent the model not only visually [118]. Nevertheless, Fidan et al. [53] already adopted the model to represent variability for frugal production aiming to support sustainable production regarding the SDGs. Furthermore, the ideas concerning an integrated reuse and variability modeling approach [118] have provided foundational knowledge for the Christian Doppler Laboratory For Mastering Variability in Software-intensive Cyber-Physical Production Systems (CDL VaSiCS)[1] [157] resulting in a long-term collaboration with JKU Linz. Furthermore, its resulted in the *Variability Modeling Body of Knowledge (VMBoK)*.[2]

**Future Work** in this direction concerns *further raising the awareness for SPL methods and disseminating the results of the thesis*, in particular, on integrated PPR variability with process sequences, to a broader industrial audience. Developing easy-to-understand educational material and engaging in workshops or industrial focus groups, such as, INCOSE MBSE or INCOSE SPL.[3]

**Production Knowledge Model and Reuse and Variability Management.** Further, this thesis has introduced the PPR *production knowledge model* and PPR–DSL [123, 130]. This approach allows human- and machine-readable modeling of product, production process, and production resource elements. Furthermore, the PPR–DSL supports the explicit representation of variability, enabling engineers to capture this knowledge systematically. Our industry partners utilize the ideas for designing their strategic CPPS engineering information ecosystem. The feedback was generally positive with an “easy to use” PPR–DSL “once the syntax is clear” and the PPR–DSL being “great because it is not as complex as, e.g., SysML.” [137]

However, our industry partners and (senior) domain subjects pointed out several limitations from a user study, demanding further improvements, for instance, due to “a steep learning curve.” [137] These demands range from a “simplified syntax to be usable for engineers” as it is “sometimes redundant and partially confusing”, “redundancies that should be omitted”, to “improving the documentation”. Furthermore, it requires “additional and better tool support to use it efficiently” with “better tool feedback” and “a better overview of PPR concepts” using, for example, “low code approaches” [137].

[1]CDL VaSiCS: `https://www.jku.at/cdl-vasics/`
[2]VMBoK: `https://github.com/SECPS/VMBoK`
[3]International Council on Systems Engineering (INCOSE): `https://www.incose.org`

Beyond the industrial collaboration, the production knowledge model and PPR–DSL have been picked up as the foundation for research in our group [169, 172] and a maturity evaluation of DSL ecosystems in a transnational research collaboration [64]. Furthermore, the approach was picked up for a visual representation of the PPR models in a line of master theses [35, 153, 100, 21] to foster the low code approach. However, the visual representation in combination with the PPR–DSL is not in the scope of this thesis.

**Future Work** in this direction concerns further advancing the PPR–DSL and its tool support. In particular, it includes reevaluating the ongoing work on simplification and tool support. Such a reevaluation should be conducted in more extensive empirical evaluations, such as usability and usefulness, with our industry partners and industrial practitioners.

For the *IRVM* research line, this thesis has adopted the PPR–DSL model for transforming those models into state-of-the-art variability models. This includes providing four openly available, reusable real-world case studies [127] in the sense of open science and dissemination. Furthermore, the PPR–DSL was adopted in the IPSE [132] and EIPSE [137] approaches. These approaches provide structured methodologies for integrating products and their variability with the corresponding production process and resource variability as required for CPPS families [55, 147]. These approaches have leveraged feature and decision models to handle structural and behavioral variability, addressing a significant gap in traditional variability management techniques [59, 174]. An essential contribution of the thesis has been the EIPSE toolchain built on state-of-the-art software. It automates critical steps in the CPPS configuration process, including transforming PPR–DSL models into variability models [43] and configuring these models to generate concrete CPPS solutions. This toolchain has enhanced engineers' ability to manage CPPS lines efficiently, reducing the manual effort and potential for errors associated with traditional approaches. A feasibility study of four real-world case studies [127] and a user study with (senior) domain experts have validated the practical applicability in an additional case study [137]. These studies have provided evidence that IRVM strategies improve engineering efficiency and facilitate better knowledge exchange across disciplines. These are believed to ultimately lead to higher quality and more adaptable CPPS solutions.

The feedback noted that "the process digitalization is a great idea that can improve reuse of existing configurations", "makes the knowledge about the production sequence explicit", and ' "supports the reproducibility of process selection." [137] Furthermore, the feedback confirmed the separation of concerns through "modeling relations from different discipline perspectives." However, the domain experts also noted that "the toolchain requires better integration" and that "executing the process using the EIPSE toolchain requires a lot of preparatory steps, which could be reduced." One expert also mentioned that "the often rigid integration structures of large companies might render the approach better suited for small and medium companies."

For ACEA, this thesis has presented the CSR method [135] that provides a framework to introduce DAE based on CSBE. The framework describes the activities to elicit CPPS knowledge from prior projects and model them as reusable artifacts PPR and C&Ss

models. This aims at improving the loose coupling of production processes and resources for a common artifact repository. Furthermore, the framework presents activities on how to use, configure, and test these reusable artifacts in particular engineering projects.

The research on the PPR knowledge model and PPR–DSL, the EIPSE, and CSR approaches involved several industrial case studies and evaluations with domain experts. However, these evaluations were quite limited in the number of participants and the size of the case studies. Furthermore, while, for instance, the EIPSE approach indicates a certain efficiency improvement due to the fast configuration of the interdependent variability models after changes to the PPR models [137], this needs additional proof.

**Future Work** in this direction concerns *further empirical evaluations*. Such empirical evaluations should be conducted on a broader real-world case study. Furthermore, they should include more engineers who represent the different disciplines involved in the engineering process, i.e., product designers, process engineers, and, for instance, mechanical engineers. To ensure the rigor of such an evaluation, guidelines for systematic and empirical evaluation should be considered [177, 205, 214, 52].

Similar to the evaluation with engineers, an *evaluation of the scalability of the approaches* needs to be conducted. This concerns, for instance, the deployment and retrieval of the PPR models and libraries in a common artifact repository to support the EIPSE approach and toolchain and the CSR framework. From the software engineering perspective, tools like Maven[4] or npm[5] for engineering build could be a blueprint. As deployment platforms, similar approaches like the Docker Hub[6] could be used.

Furthermore, the CSR and EIPSE approach need to be developed toward a higher level of technical readiness, including the functionality and *the further integration of the toolchain* in the CPPS engineering context.

**Adoption in the Industry.** Adopting the proposed approach in industrial contexts presents several opportunities and challenges. A first step toward adoption must be, as mentioned above, *raising the awareness* for SPL engineering in general and for the concepts of the proposed approaches in particular. For instance, the SPL community reaches out, for instance, with the "product line hall of fame".[7] However, the CPPS research and engineering domain is currently undergoing the fourth and fifth revolution, including the recent developments in artificial intelligence, which binds resources at other domains.

Due to similar reasons, several of the following points must apply to implement SPL approaches, such as ours. We argue that there needs to be certain pain points that can be addressed through the approach. In our case, *we deem it the required flexibility and a current change to brownfield engineering*, which requires improved reconfigurability

[4]Maven: `https://maven.apache.org`
[5]NPMJS: `https://www.npmjs.com/`
[6]Docker Hub: `https://hub.docker.com/`
[7]Product Line Hall of Fame: `https://splc.net/fame.html`

of CPPSs. Additionally, there needs to be strong support from the CPPS engineering organization's management and a collaboration champion within the CPPS engineering organization [213]. Similarly, there need to be short-term results that have an impact [213] on the CPPS engineering organization and the involved domain experts. These are, for instance, avoiding semantic inconsistencies and overcoming *clone-and-own* approaches.

Therefore, there needs to be the provision of well-documented *comprehensive and solid tool prototypes* for the proposed approaches tested, at least, on CPPSs of lab size as showcases. These prototypes should support the different concerns of the involved engineering disciplines and minimize manual effort and complexity. This is because it is a significant monetary effort for a CPPS engineering organization to pull off high-valuable engineers from their daily business to participate in activities, such as testing software, as our industry partners confirm. This goes hand in hand with the multidisciplinary domain, as CPPS engineering is a multidisciplinary effort, and testing would require several engineers of different disciplines. This might be countered by splitting the tests into domain-specific tests and then testing the toolchain in integrated multidisciplinary tests.

However, testing the approaches and practices would require the investment of upfront efforts. This includes, for instance, the preparation of particular models, filling engineering libraries with potential production processes and resources, and training of domain experts to learn the PPR–DSL and EIPSE in a tool environment that is not yet integrated into their software ecosystem. Short-term benefits could be eliciting engineering knowledge from domain experts for integrated PPR aspects or the enhanced reproducibility of the configuration approach. Nevertheless, the EIPSE approach is quite a complex approach that might induce more overhead than introducing isolated measures, which needs to be investigated further.

Nevertheless, such upfront efforts require a thorough analysis of the costs and benefits to demonstrate the long-term benefits of our approaches in CPPS engineering. This includes investigating, for example, reduced errors, increased engineering efficiency and quality, and cost reduction during basic planning. While there are cost models for SPL engineering, such as COCOMO [152, 197], to our knowledge, such a cost-benefit approach has not been developed for multidisciplinary engineering, including CPPS engineering. Finally, it requires developing an approach for conducting incremental migrations toward CPPS family engineering in several stages, backed by evidence and experience from the practice. A blueprint could be, for instance, similar to what Grüner et al. [67] proposed for SPL engineering in robot automation.

**Future Work** that can drive this direction further include, among other things, the following. Additionally, workshops with our industry partners who are now aware of our proposed approaches but who require support on how to integrate them into their engineering practice and software ecosystem. Advancing the approaches and prototypes toward a point where engineers of different disciplines can use them easily. Investigation of the costs and benefits, for instance when implementing the approach in a lighthouse project with interested CPPS domain experts.

**Production Reference Model** Finally, the thesis has extended the PPR production knowledge model toward *ACEA*. To this end, it provides the I4AN knowledge model and the I4AN reference model [128, 129] as a foundation for the identification of CPPS engineering design patterns. The reference model and the engineering design patterns have fostered coordination and communication of multidisciplinary knowledge in CPPS engineering. The approach has provided the foundation for research on coordinated CPPS reuse for multidisciplinary CPPSs engineering knowledge [134], publications in our research group [13, 14, 16, 168] and collaborations [76, 112]. Beyond that the approach has been successfully applied and been well received in a corporation with a German automotive manufacturer for advanced data analytics [77] Additionally, it has been successfully applied and been well received in a corporation with an Austrian supplier of automotive parts for production quality improvement [102, 103].

**Future Work** in this direction could be the development of support to conduct the approach better.

**Takeaways for the SPL community.** Considering the application of SPL approaches by CPPS engineering, the SPL community may take away the following points. *First*, while engineering product lines solely for software is already a challenge, implementing these approaches in a multidisciplinary domain presents even greater difficulties. In such a domain, multiple heterogeneous engineering disciplines must collaborate on a highly complex system, further challenging current methods and techniques. *Second*, in contrast to SPL engineering, in CPPS engineering, the identification of particular variants is far more critical. This significance arises because unnecessary product variants do not contribute to the "return on investment" of a CPPS. Even worse, planning for manufacturing product variants that do not contribute can lead to a significant overshoot in production cost, for instance, if resources are not working on capacity or are not even used. *Third*, the physical component of CPPS must be considered more thoroughly. This concerns, in particular, the testing and verification of approaches in the real world. Setting up test environments and gracefully shutting them down includes considerably more effort and cost than for sole SPLs.

Beyond the thesis's publications, the research project has led to several activities in the scientific community. One is a transnational collaboration on CSBE resulting in further publications [58, 97] and an industry whitepaper[8] and frequent exchanges with European research groups. Additionally, for dissemination, the author and a fellow group of researchers have successfully organized several editions of the special session on *Capability and Skill-based Engineering of Manufacturing Systems* at the international conference on *Emerging Technologies and Factory Automation*. In addition, the author was invited to organize several editions of the workshop on *Variability and Evolution of Software-intensive Systems* at the international *Systems and Software Product Line Conference*.[9]

[8]Platform 4.0 C&S Whitepaper: `https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/CapabilitiesSkillsServices.html`

[9]SPLC 2023: `https://2023.splc.net`

Furthermore, for dissemination, we have successfully organized two special sessions on *Software Engineering for Cyber-Physical Production Systems* at the international conference on *Emerging Technologies and Factory Automation*[10], resulting from the workshop on *Software Engineering in Cyber-Physical Production Systems.*[11]

## 4.2 Conclusion

This cumulative PhD thesis presented methods, models, and techniques for efficient reuse and variability management for Cyber-Physical Production System (CPPS) families.

The thesis addressed essential challenges in CPPS engineering including discipline-specific implicit CPPS engineering knowledge and a mainly manual engineering process that is inefficient and hard to reproduce. An additional challenge is the lack of effective and efficient engineering knowledge representation with variability. This variability includes the primary dimensions of variability in CPPSs, i.e., products, processes, and resources, required to efficiently build CPPS families. These challenges have been particularly prominent in multidisciplinary environments where knowledge communication, consistency, and reproducibility are essential for efficient CPPS design and operation. At the same time, separating the discipline-specific concerns is crucial for agile CPPS engineering.

The thesis has been organized along three main research lines: **RL1**. the *Production Knowledge Models (PKMO)* for multidisciplinary CPPS engineering knowledge representation with variability. The research line is focused on making implicit engineering knowledge explicit and mitigating issues that arise from heterogeneous engineering artifacts and unclear semantics. These efforts aim to facilitate multidisciplinary engineering knowledge modeling and exchange (cf. Section 3.1); **RL2**. the *Advanced CPPS Engineering Applications (ACEA)* for improved engineering coordination and engineering knowledge communication. These efforts aimed to provide the foundations for engineers to reuse engineering knowledge and models systematically (cf. Section 3.2); and **RL3**. the *Integrated Reuse and Variability Management (IRVM) for CPPS engineering* for semi-automated, systematic, and integrated management of the variability of reusable CPPS family artifacts (cf. Section 3.3). This research line aimed to facilitate reuse and variability management compared to a traditional manual approach to enhance the efficiency of CPPS family design (cf. Section 1.1.2).

The thesis project contributed ten core publications (cf. Table 2.1) to this cumulative thesis and several, mostly related, additional publications (cf. Table 5.1). From these publications, the thesis presented the following contributions. In the context of *PKMO*, we presented the superimposed Product-Process-Resource (PPR) model that serves as a visual representation of PPR with variability. In an evaluation with the domain experts the superimposed PPR model was positively received. However, they also demanded

[10]ETFA 2024: `https://2024.ieee-etfa.org`
[11]`https://rickrabiser.github.io/secpps-ws/`

additional tool support and textual presentations to counter visual complexity. As a result, we developed and discussed the PPR meta-model as a formal model-based representation with variability and the Product-Process-Resource Domain-Specific Language (PPR–DSL) to make implicit engineering knowledge explicit. Evaluations with domain experts showed the benefits but also led to improvements, such as a simplified syntax and additional tool support in ongoing work. For *ACEA*, we presented the Industry 4.0 Asset Network (I4AN) meta-model and the I4AN reference model for improved coordination and communication of engineering knowledge. Furthermore, we introduced basic engineering design patterns to address common engineering problems. The I4AN reference model was evaluated and already applied in industrial practice for engineering knowledge communication. Additionally, we presented the Capability and Skill Reuse (CSR) framework to reuse engineering knowledge systematically. In the context of *IRVM for CPPS engineering*, we presented four real-world case studies with PPR models and transformations to state-of-the-art variability models and published them with the engineering artifacts in the sense of open science. Furthermore, we introduced the systematic, semi-automated, and integrated Extended Iterative Process Sequence Exploration (EIPSE) approach that extends our Iterative Process Sequence Exploration (IPSE) approach. EIPSE aimed to manage the three primary dimensions of variability that PPRs spawned in CPPS families. To evaluate the approach we conducted a user study with several (senior) domain experts.

The findings showed that the *PKMO* enable the straightforward externalization of engineering knowledge with variability. The *ACEA* demonstrated an improved multidisciplinary knowledge communication in industrial practice and increased reuse of engineering artifacts. The EIPSE allows an efficient represent CPPS engineering knowledge in verifiable variability models. Furthermore, the evaluation of the EIPSE with domain experts implies that the approach allows the efficient configuration of the three primary dimensions of CPPS variability. As a result, the approach enables reusing engineering knowledge and artifacts and managing their variability efficiently. These findings, we argue, indicate that the approach can reduce the effort required from domain experts when evolving or designing CPPSs families.

In a discussion, we detailed how the approaches can benefit the environment. However, we also raised limitations concerning the approaches and sketched necessary future work to adopt the approaches in the industry.

This thesis has made significant contributions to the relevant stakeholder groups. Several concepts of the presented approaches were picked up and applied by our industry partners in the research environment. We contributed additions to the CPPS research community, for instance, by providing methods and models for externalizing engineering knowledge with variability based on standards and guidelines. Furthermore, we presented frameworks for reuse and variability management of the primary aspects of CPPS engineering, i.e., PPR models. We contributed foundational research for the Software Product Line (SPL) community concerning the connection of structural and behavioral variability. Furthermore, we presented crucial insights into CPPS engineering relevant for adopting SPL methodologies and frameworks in this field.

# CHAPTER 5

# Publications

The following peer-reviewed publications, grouped according to the research lines, contributed to this cumulative thesis. We provide detailed information on the particular publication in a corresponding section. This information comprises the publication's full citation, its aim in the context of this thesis, and the abstract. In addition, we include the contributions to the thesis concerning the thesis' goals (cf. Section 1.3) and research questions (cf. Section 1.4.5).

In Section 5.4, Table 5.1 chronologically lists additional publications of the thesis. However, several of these publications either provide building blocks for the core publications or pick up the concepts developed in the core publications of this thesis. Finally, Table 5.2 shows currently planned publications.

## 5.1 Production Knowledge Models

### 5.1.1 Towards modeling variability of PPR in CPPS engineering

#### Citation

[118] **K. Meixner**, R. Rabiser, and S. Biffl. Towards modeling variability of products, processes and resources in cyber-physical production systems engineering. In C. Cetina, O. Diaz, L. Duchien, M. Huchard, R. Rabiser, C. Salinesi, C. Seidl, X. Tërnava, L. Teixeira, T. Thuem, and T. Ziadi, editors, *Proceedings of the 23rd International Systems and Software Product Line Conference, SPLC 2019, Volume B, Paris, France, September 9-13, 2019*, volume B, pages 68:1–68:8. Association for Computing Machinery, 2019. doi: 10.1145/3307630.3342411

### Aim

The publication “Towards modeling variability of products, processes, and resources in cyber-physical production systems engineering” [118] conveys the first idea of integrating state-of-the-art PPR models and variability modeling for improving CPPS design reuse and flexibility. The publication (i) elicits challenges of the research topic from domain experts, (ii) provides a running real-world example, (iii) introduces superimposed CPPS engineering model as a knowledge model that integrates variability modeling, (iv) contributes a draft for a potential prototypical software architecture, and (v) sketches a basic research agenda.

### Contribution to the thesis

This publication contributes CPPS challenges, research vision, and research agenda, and to the research goals IRVM idea (**G2**.), superimposed CPPS knowledge model (**G1**.), and IRVM prototype architecture (**G4**.).

This publication contributes to the research questions **RQ1**., **RQ2**., and **RQ3**. by addressing the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of interdisciplinary collaboration* C5..

### Abstract

Planning and developing CPPSs are multi-disciplinary engineering activities that rely on effective and efficient knowledge exchange for better collaboration between engineers of different disciplines. The PPR approach allows modeling products produced by industrial processes using specific production resources. In practice, a CPPS manufactures a portfolio of product type variants, i.e., a product line. Therefore, engineers need to create and maintain several PPR models to cover PPR variants and their evolving versions. In this paper, we detail a representative use case, identify challenges for using Variability Modeling (VM) methods to describe and manage PPR variants and present a first solution approach based on cooperation with domain experts at an industry partner, a system integrator of automation for high-performance CPPS. We conclude that integrating basic variability concepts into PPR models is a promising first step and describe our further research plans to support PPR VM in CPPS.

# Towards Modeling Variability of Products, Processes and Resources in Cyber-Physical Production Systems Engineering

Kristof Meixner
Christian Doppler Lab CDL-SQI, ISE
TU Wien, Austria
kristof.meixner@tuwien.ac.at

Rick Rabiser
Christian Doppler Lab MEVSS, ISSE
Johannes Kepler Univ. Linz, Austria
rick.rabiser@jku.at

Stefan Biffl
Inst. of Information Systems Eng.
TU Wien, Austria
stefan.biffl@tuwien.ac.at

## ABSTRACT

Planning and developing *Cyber-Physical Production Systems* (CPPS) are multi-disciplinary engineering activities that rely on effective and efficient knowledge exchange for better collaboration between engineers of different disciplines. The *Product-Process-Resource* (PPR) approach allows modeling products produced by industrial processes using specific production resources. In practice, a CPPS manufactures a portfolio of product type variants, i.e., a product line. Therefore, engineers need to create and maintain several PPR models to cover PPR variants and their evolving versions. In this paper, we detail a representative use case, identify challenges for using *Variability Modeling* (VM) methods to describe and manage PPR variants, and present a first solution approach based on cooperation with domain experts at an industry partner, a system integrator of automation for high-performance CPPS. We conclude that integrating basic variability concepts into PPR models is a promising first step and describe our further research plans to support PPR VM in CPPS.

## CCS CONCEPTS

• **Software and its engineering → Software product lines**.

## KEYWORDS

Variability Modelling, Product-Process-Resource, Cyber-Physical Production System

**ACM Reference Format:**
Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2019. Towards Modeling Variability of Products, Processes and Resources in Cyber-Physical Production Systems Engineering. In *23rd International Systems and Software Product Line Conference - Volume B (SPLC '19), September 9–13, 2019, Paris, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3307630.3342411

## 1 INTRODUCTION

In recent years, computation and communication technologies increasingly collaborate with connected smart physical devices, building ubiquitous *Cyber-Physical Systems (CPSs)*, which are capable of autonomously interacting with their environments, including humans [33, 41]. *Cyber-Physical Production Systems (CPPSs)*, such as automated car manufacturing plants or steel mills, reflect the characteristics of *CPSs* to industrialized manufacturing [6, 33]. Planning and developing successful *CPPSs* are engineering tasks requiring a multi-disciplinary team effort of engineers from different domains, such as mechanical, electrical and software engineering [6, 8]. Innately, the involved disciplines establish different mindsets resulting in a variety of heterogeneous engineering artifacts. Therefore, in multi-disciplinary teams a proficient knowledge exchange is crucial for an effective and efficient collaboration, subsequently requiring an adequate knowledge representation, which often does not exist.

Model-based engineering artifact representations can help to bridge the gap of engineering knowledge transfer [4, 59] as they are easy to exchange among domains and can be transformed to represent relevant concepts of other domains. The *Product-Process-Resource (PPR)* approach [48], for example, allows defining relations between products to produce (e.g., a cake or rocker switch), the associated production processes (e.g., assembling or welding), and the necessary production resources (e.g., machines or robots). The *Formal Process Description (FPD)* [56] is a formal notation to realize *PPR* by modeling production processes with their input, intermediate, and output products as well as the resources needed in the processes to manipulate these products. Such concepts allow modeling, e.g., *assembly sequences*, which define the manufacturing steps for a particular product variant as a basis for designing the layout of the *CPPS* and describing/optimizing production plans, easily.

However, (a) approaches, such as the *FPD*, are still limited for modeling variability of the involved artifacts and (b) existing *Variability Modeling (VM)* approaches [11] might not be able to deal with the variability of processes, products, as well as resources in a *CPPS* context. Existing work from the area of *Software Product Line (SPL)* engineering either addressed (software) product variability [2, 38, 55] OR process variability [45, 52], but not their combination. Integrated software process and product lines have been proposed as a vision [43], but, at least in the *CPPS* context, this vision has not been achieved. Further, production resources have been only indirectly addressed in work on non-functional properties and product lines [51].

A combined approach for *PPR Variability Modeling* (*VM*) seems to be not readily available. While a multi-product line approach [21] might seem promising on the first glance, we will show that product, process, and resource variability should be modeled in an integrated manner, and not as separate product lines. Finally, in a *CPPS* context, integrated *PPR* modeling is not the only challenge for a *VM* approach, primarily due to the heterogeneity of artifacts and involved domains, and due to the continuous and independent evolution of products, processes, and resources.

In this paper, we identify and detail the challenges for *VM* approaches in the *CPPS* context. Furthermore, we evaluate a first *PPR* notation extension for variability in a case study with domain experts at an industry partner and present a research agenda directed towards achieving the goal of modeling the *PPR* variability in *CPPSs*.

The remainder of this paper is structured as follows. Section 2 presents the background of *CPPS* and *PPR* and discusses related work on *PPR* and *VM*. Section 3 describes our research questions. Section 4 presents an illustrative use case, which is used to derive challenges for *VM* in *CPPS* presented in Section 5. Section 6 presents a first solution approach and describes a research agenda. We conclude the paper in Section 7.

## 2 BACKGROUND AND RELATED WORK

This section summarizes related work on *CPPSs*, *PPR*, and *VM*.

### 2.1 Cyber-Physical Production Systems

The scientific community uses several definitions of *CPSs*, which Gunes et al. [20] summarize in a survey on *CPSs* concepts and challenges as *"complex, multi-disciplinary, physically-aware next-generation engineered systems that integrate embedded computing technology into the physical phenomena"*. Examples range from home automation solutions based, e.g., on Google Echo, over truck fleet logistics using GPS for real-time coordination, to large-scale infrastructure like smart grids allocating resources depending on energy load. Key capabilities of *CPSs* are, e.g., robustness, safety, and adaptation to environmental characteristics in real-time through, e.g., self-diagnosis, self-adaptation, and self-maintenance [33, 41]. Krüger et al. [27] state that a central challenge of *CPSs* stems from managing the variability of their heterogeneous aspects.

In this paper, we utilize [6] that defines *CPPSs* as advanced production systems building the foundation for the 4th industrial revolution [33]. Using smart physical infrastructures, the latest data, computer, and communication technology, as well as modern production methods, like additive manufacturing, *CPPSs* facilitate optimized production processes along the value chain for a variety of products with a broad range of characteristics.

As mentioned *CPPS* engineering requires efforts of engineers from various domains, building a multi-disciplinary environment [6]. Furthermore, engineers often build groups, e.g., basic vs. detailed planning, depending on the process phase [23]. On top, they consult experts with cross-cutting knowledge, like safety and security, for support. In such settings, disciplines employ different paradigms and use heterogeneous engineering artifacts, technologies, and tools [34]. For instance, electrical engineers use wiring plans as *CPPS* perspective and specific tools to manipulate them. Therefore, engineers working together necessarily establish auxiliary common concepts [35] that describe similar elements of *CPPSs* and act as interfaces between domains. For example, in wiring plans control lines for sensor inputs are modeled and connected via wiring to robot sensors that are familiar to mechanical engineers. However, due to the varying *CPPS* perspectives and the heterogeneous artifacts, auxiliary concepts are often insufficient. Aiming at enabling more effective and efficient knowledge exchange, it is, therefore, crucial to establish representations that include relevant *CPPSs* aspects and their commonalities and variability.

### 2.2 Product-Process-Resource

*CPPS* engineering is often optimized for *intra-disciplinary* processes [23]. Still, the heterogeneity and incompatibility of paradigms, domain-specific tools, and artifacts in the *interdisciplinary* knowledge exchange may result in low-quality data and, subsequently, in planning errors.

On top, the internal semantics of engineering artifacts, such as *Excel* spreadsheets, which are frequently used, often require expert interpretation. Therefore, model-based, machine-readable, and easily exchangeable engineering representations are the foundation for bridging gaps in the knowledge transfer [4] by providing common concepts [35] between engineering domains. However, as the purpose of *CPPSs* is to manufacture products by combining production resources in production processes [15], these varying aspects are inseparably linked. This implies the need for comprehensive models to express the requirements of products towards *CPPSs*.

Schleipen et al. [48] coined the term *PPR model* based on the trinity of (a) the product with its characteristics and components (Bill of Materials), (b) the processes with their characteristics (Bill of Operation), and (c) the resources executing the processes. In the *PPR* model, these aspects of a *CPPS* are treated as first-class objects.

Several approaches support modeling *PPR* concepts, like the initially proposed approach of Schleipen et al. [48], which heavily builds upon *AutomationML*[1] and the *ISA 95* [22], which indirectly allows the representation of *PPR* but is more oriented at batch processing and the description of interfaces between manufacturing information systems. We decided to use the *FPD* approach [56] that, in contrast to the before mentioned approaches, provides a direct, tool- and technology-agnostic way to represent *PPR* concepts. The *FPD* defines a graphical notation and a data model for modeling connected production processes, including their boundaries that accept process input products and yield output products by employing specific resources (see left side of Figure 1 for an example).

### 2.3 Variability Modeling

There is a plethora of *systematic (literature) reviews (SLRs)*, mapping studies, and surveys that discuss *Variability Modeling* (*VM*) approaches [3, 5, 10, 11, 18, 39]. For example, in a tertiary study Raatikainen et al. [39] investigate structured reviews in the field of software product lines to extract (a) how software variability is modeled, (b) which kinds of variability models exist, and (c) which level of evidence was found in the reviews. Their findings show that research on the variability of process models and quality attributes is underrepresented. Rather than new approaches of *VM*, they advocate the adaptation and combination of existing approaches to particular problem contexts to make *VM* applicable for industry.

Galster et al. [18] provide an *SLR* investigating current trends of *VM* and how variability is handled in software engineering, but also gaps in the research area. Their results show that only a few works consider business processes as artifacts in *VM* and that design-time qualities, including evolvability, has been less addressed in existing research.

There is some work focusing on *process variability/software process lines*. For example, Rosa et al. [44] report on a survey on *VM* for business processes. They found variability mostly modeled by

[1] AutomationML - https://www.automationml.org/

extending existing business process models with *VM* elements, e.g., by exchanging process tasks for sub-process templates. Classical *VM* techniques, such as feature or decision models, were more seen as decision support during customization when choosing elements to add to the business process models. Simmonds et al. [52] describe their experiences of using a mega-modeling approach to define processes and their variability. Rouillé et al. [45] propose an approach to apply the *Common Variability Language* to model requirements variability and their relation to development processes. Lamprecht et al. [28] present a behavior-oriented approach to variability management for supporting process developers in selecting processes that match service constraints.

There is also some work on *non-functional properties and variability*, which might be useful to deal with resource variability [51]. Sincero et al. [53] describe their *Feedback Approach*, which extends traditional *SPL* engineering to improve the configuration of non-functional properties. Ghezzi and Sharifloo [19] explain how probabilistic model checking techniques and tools can help verify non-functional properties of configurations derived from a *SPL*.

There is a rich body of work on modeling *CPSs* [12, 29], e.g., modeling architecture and behavior [42] or modeling system goals of self-adaptive systems. Modeling *CPSs* clearly has been recognized as an important approach to deal with their complexity and heterogeneity [29]. Variability, however, has not been the scope of these works. Except for some initial works [27] that mainly discuss challenges and potential solutions, to the best of our knowledge, there is no approach explicitly and systematically tackling the variability of products, processes, and resources in an integrated manner in the context of *CPPSs*. Existing work from the *SPL* community, e.g., on partial models or perspectives on feature models [26, 49] and on supporting interdisciplinary product lines [16, 24], needs to be adapted and extended for this context.

## 3 RESEARCH QUESTIONS

This section raises research questions that we identified from related work and discussions with domain experts at an industry partner – a provider of automation solutions for high-performance *CPPSs*.

***RQ1**: Which aspects of CPPS engineering mainly challenge the capabilities of existing variability modeling methods? CPPS* engineering is embedded in a multi-disciplinary environment with diverging characteristics to software engineering, where *SPL* engineering stems from, and traditional manufacturing. To enable *VM* in the context of *CPPS* engineering in practice, we need to identify which particular aspects of such projects challenge existing *VM* approaches. We address this *RQ* by analyzing the literature from Section 2 and interviewing domain experts of our industry partner. By answering *RQ1*, we aim to sketch and investigate possible research directions to address the identified challenges and adapt existing *VM* approaches for use in the *CPPS* context.

***RQ2**: How can a PPR model, such as the FPD, be extended to represent both product and process variability as foundation for co-evolution of products and processes in the design of assembly sequences in CPPS engineering? FPD* is a standard approach that reflects *PPR* aspects of *CPPS* engineering. However, the approach does not take *VM* aspects into account, e.g., to model the commonalities and variability of products and related manufacturing processes. Therefore, we investigate how extending the *FPD* modeling notation can represent both solution space variability and problem space variability to domain experts as a foundation for designing co-evolution of products and processes in the design of assembly sequences in *CPPS* engineering. A first goal is enabling engineers to model variability in the *assembly sequence* efficiently based on a suitable notation as a fully combined variability model is likely to overwhelm them. Despite this goal we need to further investigate how to combine the *PPR* notation with advanced aspects of *VM*.

## 4 ILLUSTRATIVE USE CASE

We present the *cake baking* use case, based on [60], to illustrate the variability in *CPPSs* and challenges for existing approaches.

Figure 1 shows, on the left-hand side, a part of a *FPD* modeling a *cake baking production process* along with its products and resources, and, on the right-hand side, corresponding orthogonal feature models of the particular *PPR* aspects. Exemplary variation points from the *FPD* model, indicated with gray circles and an ID *VarX*, are reflected in the orthogonal feature models.

A cake consists of a varying number of layers, a filling between the layers and an optional gloss. The layers require three input factors, i.e., *Flour*, *Milk*, and *Eggs*, depicted as circles in the figure, which together create a *Raw Dough* after a mixing process (*Mix ingredients*), shown as a rectangle, using a *Mixer*, illustrated as rounded rectangle. The dashed line around the mentioned *PPR* aspects shows the boundary of the process step. In a further process (*Bake dough*), the *Raw Dough* is then baked in an oven to create a *Cake Layer*.

Quite similar to the previous two process steps, the process *Cook filling* requires two ingredients to create a cake filling. However, in this case we can choose from alternative input factors, i.e., the *Fruit*, marked here for better readability with *Var1* in the *FPD* of Figure 1. Corresponding to the *FPD*, on the top right-hand side of the figure there is a feature model describing the *products*, i.e., the *cake* and its parts. *Fruit* are modeled as an alternative feature and marked as *Var1* in the model. Depending on the type of *Fruit* used, the *Filling* needs to be cooked at different temperatures levels. Therefore, two further variation points are highlighted in the figure. First, tagged with *Var2*, the energy of the *process*, added as *Heat*, needs to be adjusted in a certain range. Second, the *resource* used to process the *Fruit*, i.e., the *Stirrer*, labeled with *Var3* in the *FPD*, needs to fulfill the requirements of the *Filling* to be cooked. These variation points imply *cross-model dependencies* between the variability models and features that are illustrated as the red arrows. In our case, depending on the chosen feature for *Fruit*, the process requires appropriate heat implying in the *Resource Variability* feature model, to choose the *Hot Stirrer* when the cake should contain a *Strawberry Filling*.

An additional variability, indicating cross-model dependency, is the decision whether or not to put a *Gloss* on the cake (*Var4* in the figure). If the optional feature *Gloss* is not selected, the whole process (*Create gloss*) with its products and resources, indicated by the dashed boundary, has to be neglected. Further, more complex constraints that can be modeled at design time or run time could

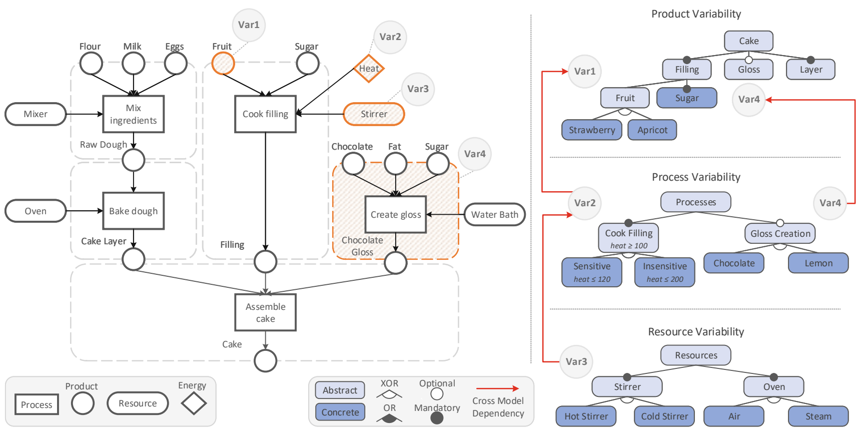


**Figure 1: *Cake baking* CPPS example represented by a *PPR* model (left) and orthogonal feature models (right) to describe Products, Processes, and Resources.**

be, e.g., the trade-off between dough creation throughput and the disintegration of the *Eggs* due to the heat generated by the *Mixer*.

This simple yet realistic *cake baking* example is an academic abstraction of our industry partner's *CPPSs* that we use to improve common understanding and due to non-disclosure agreements. It still shows the complexity that *CPPSs* can imply on variability modeling. Indeed, there is scientific evidence on the complexity of *cake baking CPPSs* in food engineering [31, 46]. One can easily transfer the example to any *CPPS* in the industrial automation domain, e.g., product aspects could be machine parts to be built, gloss could be varnish; process aspects could be the welding of machine parts (using different welding methods and heat levels) in discrete manufacturing, painting in continuous production; resources could be welding equipment or a painting robot. The cake baking example allows identifying and discussing challenges that variability modeling methods face in a *PPR*/*CPPS* context.

## 5 CHALLENGES

To answer *RQ1*, we identified the following challenges for *VM* in the context of *PPR* for *CPPSs* based on the described use case and discussions with domain experts of our industry partner.

*CH.1 – Multiple Disciplines.* In most case study reports [5, 32, 55], the modelers, who actually create variability models, typically either come from the same or similar domains, like software engineering, and/or do this with insufficient tools like *Excel*. In common practice, there is just one person responsible for the variability modeling, if there is any person. However, in *CPPS* engineering, several modelers are coming from very heterogeneous domains, such as mechanical, electrical, process, fluidic, and software engineering, with their discipline-specific tools and vocabularies. These different views imply an additional dimension in the variability models, as each of the domains brings in its own perspective on the particular *PPR* variability aspects. Existing approaches that emphasize multi-disciplinary models [1, 16, 24] could be a good starting point to develop an approach for this context.

To accurately model the variability in such heterogeneous multi-disciplinary environments, we identified three sub-challenges to address. (a) *Common concepts.* Engineers need to agree on a set of common concepts [35] defining variation points in their specific domain to later identify the connecting points representing dependencies between the variability representations. (b) *Variability dependencies.* Engineers then need to find out which changes in a discipline-specific representation need to be propagated to dependent representations to model variability consistently. For instance, safety constraints in connected variability models might be crucial for the correct operation of a *CPPS* and, therefore, are important to be preserved. (c) *Change awareness.* Newly introduced or modified concepts or variation points need to be communicated to the stakeholders of dependent domains to allow them to adapt their variability models accordingly and to ensure consistency. Challenge *CH.1* impacts several of the following identified challenges due to its cross-cutting nature.

*CH.2 – Heterogeneity.* Krüger et al. [27] affirmed that *CPSs* combine heterogeneous aspects, requiring a combined representation of the systems' variability. The authors identified three sources of heterogeneity in *CPSs*, (a) a broad range of different *artifacts*, (b) various levels of *information granularity*, and (c) mixed *representations of variability*, and proposed using either a mapping between the variability models or an integrated model for the aspects.

The *cake baking* use case demonstrates that the identified sources of heterogeneity similarly exist for *PPR* variability modeling. In our case, (a) the range of artifacts is the distinct models for the three different aspects of *PPR*, namely the product, process, and resource variability models, (b) the varying levels of information granularity can also be found in the *PPR* case as, e.g., the gloss product might be connected to more than one process step, and (c) the mixed representations that can be mapped to different modeling approaches or hierarchies of the *PPR* aspects.

Looking at Figure 1, one can tell that representations like feature models can help to structure variability but, at the same time, these models and their mutual dependencies may quickly become large, complex, and, subsequently, hard to manage for practitioners. While approaches exist to deal with product variability [11], process variability [52], and resources [51], there is no integrated approach for dealing with products, processes, and resources as well as their variability. A very specific approach that aims at reusing safety cases for safety-critical product development processes by combining these aspects was presented in [17]. For the practical *CPPS* context, it, however, remains unclear whether to prefer an integrated modeling approach or an approach with multiple separate/orthogonal models and explicit mappings.

*CH.3 – Usability.* Due to the multi-disciplinary environment, stakeholders from various domains will be involved in the variability modeling process. Therefore, the usability of the modeling tools for different types of users from the *CPPS* engineering domain is essential for a broad acceptance, which is hard to achieve when having to deal with user groups with diverse backgrounds. To this end, we also count quality in terms of, e.g., interdisciplinary correctness of the variability models, to the characteristics of usability.

Currently, variability modeling methods and tools are often academic solutions that do not have usability as their primary focus, particularly not in reported evaluations [10]. Those tools that do emphasize usability, e.g., the commercial tools *pure::variants*[2], *Gears*[3], and the (openly available) *Feature IDE*[4], have still mainly been designed envisioning users with a software (engineering) background. While some work has been investigating the usability of product configuration tools [40], to the best of our knowledge, the usability of variability modeling tools has not been considered, especially not in a *CPPS* context.

The general dilemma between automation and usability has been discussed before [37]. The key challenge in our context is to provide concepts and tools that help practitioners to model variability efficiently, even in a heterogeneous context, while maintaining the correctness at a reasonable level of complexity. Furthermore, the engineering tools of the particular domain experts have to employ the provided concepts to foster their usage in practice, while still maintaining engineering domain-specific vocabulary.

*CH.4 – (Co-)Evolution.* *CPPS* engineering is driven by requirements of industry and customers as well as by frequently evolving technologies. Therefore, the concepts represented in *PPR* variability models are subject to continuous change. The underlying *PPR* aspects, such as the production processes, are, in principle, evolutionary independent, which means that an engineer typically will change these aspects without notifying others. For instance, the DIN 8580 [14] provides a standardized catalogue for manufacturing processes that evolves independently from production resources existing on the market. However, a change in one aspect and its variability model may require the adaptation of dependent models. For instance, if a new production process is developed, variability may change not only the process variability model but also the resource variability model, if new requirements towards resources are introduced. Similarly, if, e.g., a new sort of fruit for a cake filling has to be considered, new or changed dependencies must be propagated between the product and process variability models. Remarks from engineers revealed that today, such evolutionary changes are done unsystematically in various tools and artifacts, with a propagation among domains often only on demand and, e.g., via e-mail or in informal meetings, causing issues during integration.

Such changes not only have to be considered when supporting the co-evolution of the variability models and their dependencies but also in the means to propagate such changes to participating engineers, which highly relates this challenge to *CH.1*. Existing research on the co-evolution of variability models and product line artifacts [9, 13, 30, 50] is a promising starting point to address this challenge, but, to our best knowledge and according to the reviewed literature, so far has not been applied in the *CPPS* context. Other work on co-evolution, e.g., of products, processes, and production systems in the manufacturing domain [54], has not considered variability systematically. A key goal is to achieve, at some point, an integrated, consistent variability model that can safely be used in the engineering phase. Developing a consistency checking approach [57, 58] to detect and fix inconsistencies between (models for) products, processes, and resources thus could be a promising first step to support co-evolution.

## 6 TOWARDS PPR VARIABILITY MODELING

To address the challenges described in Section 5, and following other discussions of *CPS* challenges [27] and issues regarding integrating software process and product lines [43], we aim at developing an approach to support integrated *VM* for *CPPS*. Instead of re-inventing the wheel, we build on existing *VM* approaches discussed in Section 2 and adapt and combine these approaches as needed. Promising starting points are *orthogonal VM approaches*, such as *Decision Modeling (DM)* [11] and *OVM* [38]. However, one also has to consider the availability of tools that support modeling products, processes, and resources, and their variability in an integrated manner, while supporting multi-disciplinary users, in a context with high demands on usability and continuous evolution.

### 6.1 Preliminary Industry Case Study: Assembly Sequence PPR VM for Rocker Switches

Together with domain experts at our industry partner, we conducted a case study on the usefulness and usability of means to model variability by extending the semantics of the *FPD* for *PPR* modeling. Figure 2 shows the *FPD* model extension in the context of a section of an *assembly sequence* of a *rocker switch PPR* model[5] The variants of the switches are based on real-world *CPPS* use cases from our industry partner, elicited from several documents and

[2]pure::variants - https://www.pure-systems.com/
[3]Gears - https://biglever.com/solution/gears/
[4]Feature IDE - http://www.featureide.com/

[5]*Rocker switches* have one or more rockers, e.g., realized as single-pole-single-throw, double-pole changeover, or four-way switches, controlling devices like sun-blinds.

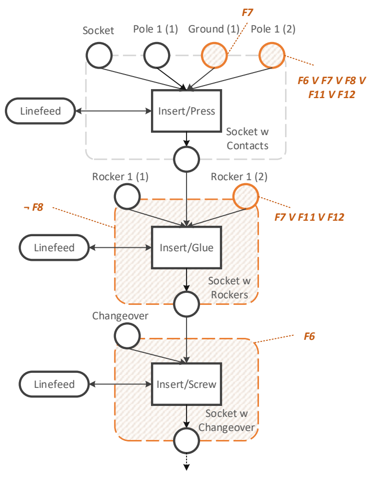


**Figure 2: *PPR* model of six *rocker switch assembly sequence* variants related with different features. Numbers in brackets indicate multiple instances of elements.**

*Excel* sheets (~300 rows × 45 columns) that engineers already use as variant matrices for products. Instead of presenting a single *PPR* model variant that engineers subsequently clone, this model represents the union of several variants, following an annotative, 150% modeling approach [47].

In the notation, we depict mandatory *PPR* elements in black, while optional/alternative elements are filled with a hatched pattern. Selection of optional/alternative[6] elements depends on the selection of certain (obfuscated) features annotated in the figure. For example, while in Figure 2 the first process is mandatory for all variants including the *Socket* and *Pole 1 (1)*, *Ground (1)* is only needed for a particular feature (F7).

When a process step is marked with a hatched pattern, all of its input factors and resources are optional. Nested optional elements are further marked with a hatched pattern. For instance, the second (insert/glue) process step in the *assembly sequence* is needed in all variants except for when *Feature 8* was chosen, as shown by the negation in the feature annotation. The second *Rocker 1 (2)* in the second process step is relevant for features *F7*, *F11* and *F12*, indicated by the pattern and the usage of the logical *OR* conjunction. This means when selecting *F7*, the optional elements *Ground (1)*, *Pole 1 (2)*, and the second process step including *Rocker 1 (2)* are needed besides the mandatory *Socket* and the *Pole 1 (1)*.

This extension of the notation addresses challenge *CH.3* as, in practice, domain experts in basic engineering first need to identify the relevant product variants from customer requirements and determine their particular *assembly sequence*. Therefore, domain experts require a *PPR* notation extension that enables them to model *assembly sequence* variants before building up the rest of the variability model. We argue that such models are useful and usable for *CPPS* engineers to express the variants of the *assembly sequence*.

[6]Note that at this point, we do not explicitly separate between optional and alternative elements in the graphical notation.

To this end, we investigated in the case study the notation extension for twelve *assembly sequences* of *rocker switch* variants. In an open interview with three domain experts at the industry partner, we discussed the model to evaluate our approach. We explained the *PPR* model notation, in particular, the extensions introduced for modeling variability. The experts provided feedback on the completeness of the model and on model expressiveness regarding relevant variants. The domain experts found the model extension to improve their way of modeling the variants of a product in comparison to their traditional way of describing variants in *Excel* spreadsheets. The domain experts also confirmed that the current version of the approach provides the required means to represent product variants properly in a single model. The experts also found the model useful to communicate their ideas to partner engineers and customer representatives. However, the experts mentioned that they would require specific tool support to design, analyze, and maintain the potentially complex model.

This preliminary case study is a first step towards answering *RQ2* (cf. Section 3) and developing an approach to support integrated variability modelling of *PPR* in a *CPPS* context.

### 6.2 Status and Research Agenda

While the examples presented in this paper allowed us to initially describe the challenges (cf. Section 5) for *VM* in the context of *PPR* for *CPPS*, to investigate *RQ1* (cf. Section 3) in sufficient detail, we need to systematically study existing *VM* approaches. We already have started conducting a *systematic mapping study* to identify and analyze *VM* approaches in literature that (a) provide at least basic support for integrated modeling of product and process variability, (b) are extensible, and (c) have the potential to be applicable to the domain of *CPPS* to (partly) address the found challenges (cf. Section 5. As there is already a plethora of systematic literature reviews and mapping studies [3, 5, 10, 11, 18], we conduct a tertiary study, similar to the study by Raatikainen et al. [39], but with a different focus: we put emphasis on finding approaches that are good in dealing with multiple disciplines (cf. *CH.1*), heterogeneity (cf. *CH.2*), and (co-)evolution (cf. *CH.4*). Tool support that has been evaluated with real users (cf. *CH.3*) will also be a major criterion.

To fully address *RQ2* (cf. Section 3), we will build on the findings of this study and on our modeling experiments with industry partners to (a) further adapt the *FPD* approach for *Variability Modeling*, e.g., by introducing abstract aspects, and (b) propose an approach for designing an *Integrated Variability Model (IVM)* for *PPR*. The *IVM* should provide a basis to model the relevant information for decision-making, such as the properties of processes and resources, and dependencies between the *PPR* aspects to build up an orthogonal variability model representing variability in the problem space. For example, a decision model could serve as a basis for detail engineering to narrow down suitable resources for a *CPPS*.

To address challenge *CH.1 – Multiple Disciplines* and challenge *CH.3 – Usability*, we plan to employ *Collective Intelligence Systems*

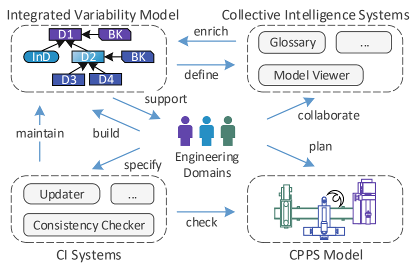


**Figure 3: Draft Solution Architecture for Integrated Variability Modelling of PPR in a CPPS context.**

*(CISs)* [36] to negotiate common concepts [35] and allow collaborative work on the *IVM* by providing discipline-specific views [25]. We have already collected engineering terms in a multi-user glossary farm[7] as a first step. To address the correctness of the *IVM* for different disciplines and to support (co-)evolution (cf. *CH.4*), we plan to apply human-supported inspection to the models [7] and adapt automated consistency checking techniques [57, 58].

Figure 3 shows a first architecture draft of our solution approach, with the *IVM* on the upper left, *CISs* examples on the upper right, a *CPPS* model on the lower right, and automated services like *Continuous Integration (CI)* systems on the lower left. Engineers from different domains are supported by the *IVM* when planning the *CPPS*. Therefore, engineering tools used by the different disciplines facilitate the *IVM* to gather relevant information concerning the perspectives of the *CPPS*. For example, when a mechanical engineer wants to decide which robot arm to use as a particular resource, a tool might provide a selection based on the properties of the product or process. The *IVM* is created and maintained by the engineers using diverse collaborative *CISs*, e.g., the glossary, that enrich the *IVM* and provide domain-specific views. Vice versa, the *IVM* defines possible selections for modeling, e.g., in a shared *PPR VM* editor. The *CI* systems check and maintain the consistency of the *IVM* and the *CPPS*, e.g., based on specific rules, and update the *IVM* if required. Such a system could, e.g., be a *CI* server that checks the consistency of the *CPPS* engineering artifacts, like electrical plans, and, in parallel, maintains the consistency of the *IVM*.

After building a first prototypical design, we will conduct further case studies with industry partners to investigate the feasibility and effectiveness of our *PPR* variability approach to address the described challenges. Based on the results of these studies, we will iteratively improve and evaluate the approach in a larger context.

## 7 CONCLUSION

When planning and engineering *CPPS*, engineers from heterogeneous disciplines have to work together, making effective and efficient knowledge exchange crucial. The *PPR* approach and the *FPD* language provide foundations for knowledge exchange in *CPPS* by representing the products to be manufactured in combination with the required production processes and resources. As *CPPSs* usually manufacture a broad product portfolio with many product variants, engineers also need to model variability in a suitable way.

In this paper, we raised *RQ1*, asking which aspects of *CPPS* mainly challenge existing *VM* approaches. We described a simple yet realistic example of a *cake-baking CPPS* to identify four key challenges for variability modeling in *CPPSs*: *multi-disciplinary views*, *heterogeneity* of artifacts, *usability* of *VM* methods/tools, and *(co-)evolution* of product and production-process variability models. We argue that existing *VM* solutions fall short in addressing (all of) these challenges and that orthogonal approaches are best suited to create a variability model that represents different views and dependencies.

*RQ2* asked how the *PPR* modeling notation can be extended as foundation for *(co-)evolution*. Interviews with domain experts revealed that first the challenge of *usability* needs to be addressed, i.e., allowing them to model variants of a basic artifact such as the *assembly sequence* of the products to build in a straightforward and integrated way. An initial evaluation of a *PPR* notation extension in a case study found potential for *PPR* modeling to help engineers describe better model variants and to explain *assembly sequence* variants to other engineers as well as customers. The *PPR* extension should serve as foundation to investigate how to tackle further challenges identified by *RQ1*. Moreover, we proposed a basic solution architecture for *VM* of *PPR* in the context of *CPPS* and a research agenda to address these challenges in the *CPPS*/*PPR* context.

As next step we plan to integrate a first orthogonal variability model describing standard processes used in manufacturing, such as the DIN 8580, to the *PPR* notation extension to further investigate the feasibility of the orthogonal approach in industrial contexts.

## ACKNOWLEDGMENTS

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

[7] Glossary - https://glossary-farm.herokuapp.com/glossaries/cdl-sqi/terms

## REFERENCES

[1] Sofia Ananieva, Matthias Kowal, Thomas Thüm, and Ina Schaefer. 2016. Implicit constraints in partial feature models. In *Proc. of the 7th Int. FOSD Workshop, FOSD@SPLASH 2016, Amsterdam, Netherlands, October 30, 2016.* 18–27.

[2] Sven Apel, Don Batory, Christian Kaestner, and Gunter Saake. 2013. *Feature-Oriented Software Development: Concepts and Implementation.* Springer.

[3] Rabih Bashroush, Muhammad Garba, Rick Rabiser, Iris Groher, and Goetz Botterweck. 2017. CASE Tool Support for Variability Management in Software Product Lines. *Comput. Surveys* 50, 1 (2017), 14:1–14:45.

[4] Luca Berardinelli, Alexandra Mazak, Oliver Alt, Manuel Wimmer, and Gerti Kappel. 2017. Model-driven systems engineering: Principles and application in the CPPS domain. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems.* Springer, 261–299.

[5] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wąsowski. 2013. A survey of variability modeling in industrial practice. In *Proc. of the 7th Int. Workshop on Variability Modelling of Software-intensive Systems.* ACM, 7–14.

[6] Stefan Biffl, Detlef Gerhard, and Arndt Lüder. 2017. Introduction to the Multi-Disciplinary Engineering for Cyber-Physical Production Systems. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems.* Springer, 1–24.

[7] Stefan Biffl, Marcos Kalinowski, and Dietmar Winkler. 2018. Towards an experiment line on software inspection with human computation. In *Proc. of the 6th Int. Workshop on Conducting Empirical Studies in Industry, CESI@ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018.* 21–24.

[8] BKCASE Editorial Board. 2017. *The Guide to the Systems Engineering Body of Knowledge.* Vol. 1.9.1. The Trustees of the Stevens Institute of Technology.

[9] Goetz Botterweck and Andreas Pleuss. 2014. Evolution of software product lines. In *Evolving Software Systems.* Springer, 265–295.

[10] Lianping Chen and Muhammad Ali Babar. 2011. A systematic review of evaluation of variability management approaches in software product lines. *IST* 53, 4 (2011), 344–362.

[11] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wasowski. 2012. Cool Features and Tough Decisions : A Comparison of Variability Modeling Approaches. In *Proc. of the 6th Int. Workshop on Variability Modelling of Software-intensive Systems*. ACM, 173–182.

[12] Patricia Derler, Edward A Lee, and Alberto Sangiovanni Vincentelli. 2012. Modeling cyber–physical systems. *Proc. of the IEEE* 100, 1 (2012), 13–28.

[13] Deepak Dhungana, Paul Grünbacher, Rick Rabiser, and Thomas Neumayer. 2010. Structuring the modeling space and supporting evolution in software product line engineering. *JSS* 83, 7 (2010), 1108–1122.

[14] DIN. 2003. DIN 8580 – Manufacturing processes. https://standards.globalspec.com/std/756719/DIN%208580 [Online; accessed 2019-04-02].

[15] Hoda A. ElMaraghy. 2009. Changing and evolving products and systems–models and enablers. In *Changeable and reconfigurable manufacturing systems*. Springer, 25–45.

[16] Stefan Feldmann, Christoph Legat, and Birgit Vogel-Heuser. 2015. Engineering support in the machine manufacturing domain through interdisciplinary product lines: An applicability analysis. *IFAC-PapersOnLine* 28, 3 (2015), 211–218.

[17] Barbara Gallina. 2015. Towards Enabling Reuse in the Context of Safety-critical Product Lines. In *Proc. of the 5th Int. Workshop on Product LinE Approaches in Software Engineering (PLEASE '15)*. IEEE Press, Piscataway, NJ, USA, 15–18.

[18] Matthias Galster, Danny Weyns, Dan Tofan, Bartosz Michalik, and Paris Avgeriou. 2014. Variability in Software Systems - A Systematic Literature Review. *IEEE TSE* 40, 3 (mar 2014), 282–306.

[19] Carlo Ghezzi and Amir Molzam Sharifloo. 2013. Model-based verification of quantitative non-functional properties for software product lines. *IST* 55, 3 (2013), 508–524.

[20] Volkan Gunes, Steffen Peter, Tony Givargis, and Frank Vahid. 2014. A survey on concepts, applications, and challenges in cyber-physical systems. *KSII Transactions on Internet & Information Systems* 8, 12 (2014).

[21] Gerald Holl, Paul Grünbacher, and Rick Rabiser. 2012. A Systematic Review and an Expert Survey on Capabilities Supporting Multi Product Lines. *IST* 54, 8 (2012), 828–852.

[22] ISA 95 2003. IEC 62264-1 Enterprise-control system integration–Part 1: Models and terminology. IEC, Genf.

[23] Lukas Kathrein, Arndt Lüder, Kristof Meixner, Dietmar Winkler, and Stefan Biffl. 2019. Production-Aware Analysis of Multi-disciplinary Systems Engineering Processes. In *Proc. of the 21st Int. Conf. on Enterprise Information Systems - Vol. 2: ICEIS,*. INSTICC, SciTePress, 48–60.

[24] Matthias Kowal, Sofia Ananieva, Thomas Thüm, and Ina Schaefer. 2017. Supporting the Development of Interdisciplinary Product Lines in the Manufacturing Domain. *IFAC-PapersOnLine* 50, 1 (2017), 4336–4341.

[25] Max E Kramer, Erik Burger, and Michael Langhammer. 2013. View-centric engineering with synchronized heterogeneous models. In *1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*. ACM, 5.

[26] Sebastian Krieter, Reimar Schröter, Thomas Thüm, Wolfram Fenske, and Gunter Saake. 2016. Comparing Algorithms for Efficient Feature-model Slicing. In *Proc. of the 20th SPLC (SPLC '16)*. ACM, New York, NY, USA, 60–64.

[27] Jacob Krüger, Sebastian Nielebock, Sebastian Krieter, Christian Diedrich, Thomas Leich, Gunter Saake, Sebastian Zug, and Frank Ortmeier. 2017. Beyond Software Product Lines: Variability Modeling in Cyber-Physical Systems. In *Proc. of the 21st SPLC - Vol. A*. ACM, New York, NY, USA, 237–241.

[28] Anna-Lena Lamprecht, Stefan Naujokat, and Ina Schaefer. 2013. Variability Management beyond Feature Models. *Computer* 46, 11 (2013), 48–54.

[29] Edward Lee. 2015. The past, present and future of cyber-physical systems: A focus on models. *Sensors* 15, 3 (2015), 4837–4869.

[30] Sascha Lity, Sophia Nahrendorf, Thomas Thüm, Christoph Seidl, and Ina Schaefer. 2018. 175% Modeling for Product-Line Evolution of Domain Artifacts. In *Proc. of the 12th VAMOS 2018, Madrid, Spain, February 7-9, 2018*, Rafael Capilla, Malte Lochau, and Lidia Fuentes (Eds.). ACM, 27–34.

[31] Mathieu Lostie, Roman Peczalski, and Julien Andrieu. 2004. Lumped model for sponge cake baking during the "rust and crumb"period. *Journal of Food Engineering* 65, 2 (2004), 281–286.

[32] Jabier Martinez, Wesley KG Assunção, and Tewfik Ziadi. 2017. ESPLA: A catalog of Extractive SPL Adoption case studies. In *Proc. of the 21st SPLC-Vol. B*. ACM, 38–41.

[33] László Monostori. 2014. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP* 17 (2014), 9–13.

[34] Richard Mordinyi and Stefan Biffl. 2015. Versioning in Cyber-physical Production System Engineering: Best-practice and Research Agenda. In *Proc. of the 1st Int. Workshop on Software Engineering for Smart CPS (SEsCPS '15)*. IEEE Press, Piscataway, NJ, USA, 44–47.

[35] Thomas Moser and Stefan Biffl. 2012. Semantic Integration of Software and Systems Engineering Environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 1 (Jan. 2012), 38–50.

[36] Juergen Musil, Angelika Musil, Danny Weyns, and Stefan Biffl. 2015. An architecture framework for collective intelligence systems. In *2015 12th Working IEEE/IFIP Conf. on Software Architecture*. IEEE, 21–30.

[37] Andreas Pleuss, Benedikt Hauptmann, Deepak Dhungana, and Goetz Botterweck. 2012. User interface engineering for software product lines: the dilemma between automation and usability. In *Proc. of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM, 25–34.

[38] Klaus Pohl, Günther Böckle, and Frank van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer.

[39] Mikko Raatikainen, Juha Tiihonen, and Tomi Männistö. 2019. Software product lines and variability modeling: A tertiary study. *JSS* 149 (2019), 485–510.

[40] Rick Rabiser, Paul Grünbacher, and Martin Lehofer. 2012. A qualitative study on user guidance capabilities in product configuration tools. In *Proc. of the 27th IEEE/ACM Int. Conf. on Automated Software Engineering*. ACM, 110–119.

[41] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. 2010. Cyber-physical systems: the next computing revolution. In *Proc. of the 47th ACM/IEEE Design Automation Conf.* IEEE, 731–736.

[42] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. 2015. Architecture and behavior modeling of cyber-physical systems with MontiArcAutomaton. *arXiv preprint arXiv:1509.04505* (2015).

[43] Dieter Rombach. 2005. Integrated software process and product lines. In *Software Process Workshop*. Springer, 83–90.

[44] Marcello La Rosa, Wil M P Van Der Aalst, Marlon Dumas, and Fredrik P Milani. 2017. Business Process Variability Modeling: A Survey. *Comput. Surveys* 50, 1 (mar 2017), 2:1–2:45.

[45] Emmanuelle Rouillé, Benoît Combemale, Olivier Barais, David Touzet, and Jean-Marc Jézéquel. 2012. Leveraging CVL to manage variability in software process lines. In *Proc. of the 2012 19th Asia-Pacific Software Engineering Conf.*, Vol. 1. IEEE, 148–157.

[46] Melike Sakin, Figen Kaymak-Ertekin, and Coskan Ilicali. 2007. Simultaneous heat and mass transfer simulation applied to convective oven cup cake baking. *Journal of Food Engineering* 83, 3 (2007), 463–474.

[47] Ina Schaefer, Rick Rabiser, Dave Clarke, Lorenzo Bettini, David Benavides, Goetz Botterweck, Animesh Pathak, Salvador Trujillo, and Karina Villela. 2012. Software diversity: state of the art and perspectives. *Software Tools for Technology Transfer* 14, 5 (2012), 477–495.

[48] Miriam Schleipen, Arndt Lüder, Olaf Sauer, Holger Flatt, and Jürgen Jasperneite. 2015. Requirements and concept for Plug-and-Work. *at-Automatisierungstechnik* 63, 10 (2015), 801–820.

[49] Julia Schroeter, Malte Lochau, and Tim Winkelmann. 2012. Multi-perspectives on Feature Models. In *Model Driven Engineering Languages and Systems*, Robert B France, Jürgen Kazmeier, Ruth Breu, and Colin Atkinson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 252–268.

[50] Christoph Seidl, Florian Heidenreich, and Uwe Aßmann. 2012. Co-evolution of models and feature mapping in software product lines. In *Proc. of the 16th SPLC-Vol. 1*. ACM, 76–85.

[51] Norbert Siegmund, Marko Rosenmueller, Christian Kästner, Paolo Giarrusso, Sven Apel, and Sergiy Kolesnikov. 2011. Scalable Prediction of Non-functional Properties in Software Product Lines. In *Proc. of the 15th SPLC*. IEEE CS, 160–169.

[52] Jocelyn Simmonds, Daniel Perovich, María Cecilia Bastarrica, and Luis Silvestre. 2015. A megamodel for software process line modeling and evolution. In *Proc. of the 2015 ACM/IEEE 18th Int. Conf. on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 406–415.

[53] Julio Sincero, Wolfgang Schroder-Preikschat, and Olaf Spinczyk. 2010. Approaching non-functional properties of software product lines: Learning from products. In *Proc. of the 2010 Asia Pacific Software Engineering Conf.* IEEE, 147–155.

[54] T Tolio, Darek Ceglarek, HA ElMaraghy, A Fischer, SJ Hu, L Laperrière, Stephen T Newman, and József Váncza. 2010. SPECIES – Co-evolution of products, processes and production systems. *CIRP annals* 59, 2 (2010), 672–693.

[55] Frank van der Linden, Klaus Schmid, and Eelco Rommes. 2007. *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering*. Springer Berlin Heidelberg.

[56] VDI/VDE 3682 2005. VDI/VDE 3682: Formalised process descriptions. Beuth Verlag.

[57] Michael Vierhauser, Paul Grünbacher, Alexander Egyed, Rick Rabiser, and Wolfgang Heider. 2010. Flexible and Scalable Consistency Checking on Product Line Variability Models. In *25th IEEE/ACM Int. Conf. on Automated Software Engineering*. ACM, 63–72.

[58] Michael Vierhauser, Paul Grünbacher, Wolfgang Heider, Gerald Holl, and Daniela Lettner. 2012. Applying a Consistency Checking Framework for Heterogeneous Models and Artifacts in Industrial Product Lines. In *Proc. of the 15th Int. ACM/IEEE Conf. on Model Driven Engineering Languages & Systems*. Springer, 531–545.

[59] Birgit Vogel-Heuser and Stefan Biffl. 2016. Cross-discipline modeling and its contribution to automation. *Automatisierungstechnik* 64, 3 (2016), 165–167.

[60] Roland Willmann. 2016. *Ontology matchmaking of product ramp-up knowledge in manufacturing industries : How to transfer a cake-baking recipe between bakeries.* Ph.D. Dissertation. TU WIen.

### 5.1.2 Integrating Variability Modeling of PPR in CPPS Engineering

#### Citation

[117] **K. Meixner**. Integrating variability modeling of products, processes, and resources in cyber-physical production systems engineering. In R. Capilla, P. Collet, P. Gazzillo, J. Krueger, R. E. Lopez-Herrejon, S. Nadi, G. Perrouin, I. Reinhartz-Berger, J. Rubin, and I. Schaefer, editors, *SPLC '20: 24th ACM International Systems and Software Product Line Conference, Montreal, Quebec, Canada, October 19-23, 2020, Volume B*, volume Part F164402-B, pages 96–103. Association for Computing Machinery, 2020. doi: 10.1145/3382026.3431247

#### Aim

The publication "Integrating Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering" [117] refines the idea of integrated PPR and variability modeling towards production process sequence variability and presents the research methodology employed for the thesis. The publication (i) presents the advanced challenges of production process sequence variability, (ii) first research questions for the thesis, and (iii) the research methodology based on design science research [73, 74, 205].

#### Contribution to the thesis

This publication contributes CPPS production process challenges, research methodology, and to the research goals IRVM (**G3**.).

This publication contributes to **RQ1**., **RQ2**., and **RQ3**. by addressing the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of interdisciplinary collaboration* C5..

#### Abstract

The Industry 4.0 initiative envisions the flexible and optimized production of customized products on CPPS that consist of subsystems coordinated to conduct complex production processes. Hence, accurate CPPS modeling requires integrating the modeling of variability for PPR aspects. Yet, current variability modeling approaches treat structural and behavioral variability separately, leading to inaccurate CPPS production models that impede CPPS engineering and optimization. This paper proposes a PhD project for integrated variability modeling of PPR aspects to improve the accuracy of production models with variability for CPPS engineers and production optimizers. The research project follows the Design Science approach aiming for the iterative design and evaluation of (a) a framework to categorize currently incomplete and scattered models and methods for PPR variability modeling as a foundation for an integrated model, and (b) a modeling

approach for more accurate integrated PPR variability modeling. The planned research will provide the SPL and CPPS engineering research communities with (a) novel models, methods, and insights on integrated PPR variability modeling, (b) open data from CPPS engineering use cases for common modeling, and (c) empirical data from field studies for shared analysis and evaluation.

# Integrating Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering

Kristof Meixner*
Christian Doppler Lab CDL-SQI, Institute of Information Systems Engineering
TU Wien, Austria
kristof.meixner@tuwien.ac.at

## ABSTRACT

The Industry 4.0 initiative envisions the flexible and optimized production of customized products on *Cyber-Physical Production Systems (CPPSs)* that consist of subsystems coordinated to conduct complex production processes. Hence, accurate CPPS modeling requires integrating the modeling of variability for *Product-Process-Resource (PPR)* aspects. Yet, current variability modeling approaches treat structural and behavioral variability separately, leading to inaccurate CPPS production models that impede CPPS engineering and optimization. This paper proposes a PhD project for integrated variability modeling of PPR aspects to improve the accuracy of production models with variability for CPPS engineers and production optimizers. The research project follows the Design Science approach aiming for the iterative design and evaluation of (a) a framework to categorize currently incomplete and scattered models and methods for PPR variability modeling as a foundation for an integrated model; and (b) a modeling approach for more accurate integrated PPR variability modeling. The planned research will provide the *Software Product Line (SPL)* and CPPS engineering research communities with (a) novel models, methods, and insights on integrated PPR variability modeling, (b) open data from CPPS engineering use cases for common modeling, and (c) empirical data from field studies for shared analysis and evaluation.

## CCS CONCEPTS

• **Software and its engineering → Software product lines**.

## KEYWORDS

Variability Modelling, Product-Process-Resource, Cyber-Physical Production System

**ACM Reference Format:**
Kristof Meixner. 2020. Integrating Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering. In *24th ACM International Systems and Software Product Line Conference - Volume B (SPLC '20), October 19–23, 2020, MONTREAL, QC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3382026.3431247

*Supervision by Stefan Biffl, ISE, TU Wien and Rick Rabiser, LIT, JKU Linz, Austria.

## 1 INTRODUCTION

The Industry 4.0 initiative[1] envisions Cyber-Physical Production Systems (CPPSs), like automated car factories, which enable the flexible and optimized production of mass-customizable products [6]. A CPPS can interact with its environment, make context-aware decisions, and self-adapt to uncertain conditions [19, 32], supporting the envisioned flexibility. Recent research shows growing interest in applying SPL engineering concepts to CPPS engineering [1, 7, 24, 30].

Key stakeholders in CPPS engineering are basic engineers and production process optimizers. *Basic engineers* design the abstract CPPS capable to produce a product line. They want to efficiently explore the large configuration space for production process design that considers product variability. *Production process optimizers* want to explore the configuration space for production process planning and scheduling. They require sufficiently accurate CPPS models to understand the consequences of their decisions regarding production process goal properties, such as throughput.

Hence, accurate models of CPPS engineering concepts are crucial for engineers to effectively and efficiently exchange their knowledge and collaborate over the engineering lifecycle [4, 46]. The *PPR* concept [41], with the *Formalised Process Description (FPD)* [45] as a formal visual model, provides a solid foundation to express CPPS engineering knowledge. However, to support the Industry 4.0 vision, such knowledge models need to include CPPS variability [30, 31], which can stem from (a) product type characteristics, (b) production process variants, and (c) CPPS resources.

There are scattered modeling approaches in CPPS engineering, like product comparison matrices, that consider aspects of variability to some extent. In addition, modeling is inaccurate, particularly regarding production process variability and characteristics due to missing dependencies to product variability. These limitations lead to the challenges of (1) weak understanding of PPR variability aspects required for CPPS design and process optimization and (2) weak understanding of the impact of changes in the product design or portfolio on CPPS production process properties. This weak understanding hampers designing CPPS process optimization methods that leverage production flexibility for a product family.

To support the required flexibility and variability, models and approaches from CPPS engineering and Variability Modeling (VM) need to be integrated. However, current research results on SPL engineering have been addressing *either* product variability [2, 33, 44] *or* process variability [39, 42]. Therefore, a sound integration requires new methods and an interdisciplinary research effort, bringing together researchers from *Model-Driven Engineering*, *Software Product Line (SPL) Engineering*, and *CPPS Engineering*.

[1]Researchagenda Industrie 4.0: https://ec.europa.eu/growth/tools-databases/dem/monitor/sites/default/files/DTM_Industrie%204.0.pdf

In preliminary research [30], we identified the challenge of heterogeneous variability representation in CPPS engineering and introduced a model-based approach to represent variability in PPR engineering models. Further, this investigation revealed the challenge of insufficient representation of variability dependencies between product and process variants in CPPS engineering.

In this paper, we motivate the need for integrated modeling of structural *and* behavioral variability in CPPS engineering employing the *Rocker Switch product line* use case. We describe our preliminary research as a basis to handle variability in CPPS engineering models, and present a first solution approach. We propose a PhD project and a corresponding research agenda on how to introduce integrated product and process variability into CPPS engineering models. The results aim at improving the accuracy of CPPS production models with variability for CPPS engineers and production optimizers.

The PhD project will follow the Design Science approach [47] resulting in the *PPR variability* framework and the *PPRVar* modeling approach. The *PPR variability* framework will provide researchers with a categorized overview on models and methods for CPPS VM and identify gaps in research. The *PPRVar* modeling approach will provide integrated PPR variability models that represent delayed design decisions in CPPS engineering more accurately. We propose a work plan, including a systematic (literature) review (SLR) and design studies on integrated PPR variability models and methods. Further, we plan case studies with CPPS engineering and optimization experts to validate the viability of the integrated PPR variability models and analyze the strengths and limitations of the designed artifacts for the next design iteration. A good research result would be a *Rocker Switch* product and process variability model that process optimizers can use to design optimized processes efficiently.

The planned research will provide the SPL and CPPS engineering research communities with (a) novel models, methods, and insights on integrated CPPS variability modeling, (b) use case data from CPPS engineering for common modeling, and (c) empirical data from field studies for shared analysis and evaluation.

The remainder of this paper is structured as follows. Section 2 discusses related work on CPPS engineering and VM. Section 3 introduces the use case *Rocker Switch product line*. Section 4 motivates research questions. Section 5 discusses the research approach and methods. Section 6 presents preliminary results and outlines a research agenda. Section 7 concludes the paper.

## 2 RELATED WORK

This section summarizes research on CPPS engineering and VM.

### 2.1 Cyber-Physical Production Systems Engineering

CPPSs are advanced production systems [6] reflecting the attributes of Cyber-Physical System (CPS) to the production domain: *complex, multidisciplinary, physically-aware, self-adaptive* systems using modern ICT to integrate their environment [13, 32]. CPPSs use modern production methods, like additive manufacturing, to facilitate optimized production processes along the value chain for product families. CPS/CPPS examples range from smart farming over automated car manufacturing plants to large-scale smart grids.

The multidisciplinary nature of CPPSs requires experts from different domains, like mechanics and electrical engineering, and cross-cutting knowledge, like safety and security, to work together [6]. Naturally, particular domains employ their own concepts, techniques, and tools that are usually optimized for *intra-disciplinary* processes, which leads to heterogeneous engineering artifacts [21]. Krüger et al. [24] identified this heterogeneity as a major challenge for managing variability. We identify **Challenge 1a**, the *weak understanding of CPPS variability dispersed in heterogeneous artifacts, often leading to an incomplete representation of variability in CPPS engineering.* It is often hard to reduce the large problem and configuration space of design options and assess their impact on production and the sequence of production steps. A foundation to bridge the gaps in knowledge transfer and to overcome the heterogeneity are shared, model-based, machine-readable, and easily exchangeable engineering representations [4], which are capable of expressing the requirements of products towards CPPSs.

Schleipen et al. [41] coined the term *PPR model* based on the three inseparably linked aspects (a) *product* with its properties and components, (b) *process* manipulating the products, and (c) *resource* that executes the processes. The initial approach of Schleipen et al. [41] focuses on batch processing and the link to manufacturing information systems. The *Formalised Process Description (FPD)* approach [45] provides a tool- and technology-agnostic representation for PPR concepts defining a formal and visual notation (see Figure 1 *Variant 1* for an example). The approach allows modeling connected production processes (rectangles) that transform input products into output products (circles) by employing production resources (rounded rectangles). Hildebrandt et al. [15] introduced the *SemAnz 4.0* approach to allow semantic reasoning by linking the FPD with a standardized, structural model [18] and a programming language [17] to express resource behavior. In Kathrein et al. [21], we extended the FPD towards *abstract* PPR aspects and *consistency expressions* to support delayed design decisions and the formulation of dependencies. Furthermore, we introduced the concept of required and provided skills and their aggregation [29], further abstracting the connection between processes and resources to facilitate the efficient matching of variable PPR aspects. To describe dependencies between processes and the processed products, which can be used during process sequence optimization, we apply pre- and postconditions [28] for processes and assembly groups. We define **Challenge 1b** for variability management in CPPS engineering as *the insufficient representation of dependencies between product and process variants and between process sequence variants.* The insufficient representation makes it difficult to define and derive valid production process solution candidates due to complex dependencies between product and process variability and the large unmapped solution space.

While these approaches are essential building blocks for CPPS VM, they focus on the resource aspect of PPR and hardly consider dependencies between product and production process variants. In this paper, we focus on representing dependencies between PPR aspects [21], in particular between product and production process variants, to facilitate CPPS engineers and process optimizers in finding valid production process solutions and in analyzing the impact of changes to product requirements on viable production process variants.

## 2.2 Variability Modeling

There is a large number of SLRs, mapping studies, and surveys that investigate Variability Modeling (VM) approaches [3, 5, 9, 10, 12, 34]. Raatikainen et al. [34] report in their tertiary study on SLRs in the area of Software Product Lines (SPLs) low representation of research on the variability of process models and quality attributes. They suggest to adapt and combine VM approaches and apply them to specific industry problems. Galster et al. [12] investigate in their SLR trends of VM, how software engineering handles variability, and gaps in the field of SPL. They show that business processes have been less addressed as artifacts in VM.

Rather few academic works focus on *Software Process Lines*. Rosa et al. [38] provide a survey on VM for business processes. Their research shows that only a few business process models have been investigated and that these are extended with VM elements, e.g., by substituting process tasks with templates. Feature or Decision Modeling was used more as a support in the business process models' customization during element selection. The authors observe that most of the approaches have neither been validated nor evaluated, requiring further research in the field. Lamprecht et al. [25] present a VM approach that supports process developers matching service constraints to process properties during their selection. Rombach [37] proposes an integration of software process and product lines as a future vision, which has not yet been achieved.

There is a plethora of research on modeling various CPS characteristics [11, 26], like the system goals, architecture, and behavior of self-adaptive systems [36]. Initial work [24] on CPSs in the context of VM discusses challenges and potential solutions. Yet, existing approaches for VM [10] seem to be incapable to handle CPS variability and the combination of PPR in CPPS engineering.

A typical approach in CPPS engineering is the use of superimposed product representatives, i.e., type representatives [14], to investigate whether the resources required for producing a superset of products were planned in a CPPS design. Yet, type representatives are usually single-use, calculated, virtual products without model and reference to the underlying features. This renders them insufficient for a systematic, model-driven approach. Interdisciplinary product lines [23] aim to model the variability of CPPS resources, maintaining a domain-specific view on the modeled variability, but do not consider products and resources. Caesar et al. [7] aim at a context-aware re-configuration of CPPSs using VM and semantic approaches. They model the variability of the CPPSs and their context in separate feature models linked through cross-tree constraints. At runtime they then adapt the CPPSs through reasoning over the problem and solution space. Yet, these approaches focus on designing resource solutions and treat processes just as requirements for resource designs. Multi-purpose, multi-level feature modeling [35] introduces a VM approach that considers different (domain-specific) views on different levels of granularity. Yet, the work does not consider the behavioral nature of production systems. Multi-product lines [16] seem promising but model product, process, and resource variability as separate product lines. Recent work on structural and behavioral variability [20] extends the domain-specific language (DSL) *CLAFER* for these purposes. We will investigate this approach's applicability to CPPS engineering, e.g., considering its usability for CPPSs engineers and domain experts.

While the related work provides a wide range of concepts and building blocks, there is no clear approach for PPR variability that *integrates structure and behavior*. Therefore, we identify **Challenge 2** as *the unclear impact of additional product variants on the variability of processes in CPPS*. The knowledge of such an impact is essential when additional products enter the product family, or additional features should be realized. To the best of our knowledge, there is no approach explicitly and systematically tackling PPR variability in an integrated manner in the CPPS engineering context. To illustrate the mentioned challenges, the next section introduces the use case *Rocker Switch product line*, abstracted from a real-world industrial use case.

## 3 THE ROCKER SWITCH PRODUCT LINE

The use case *Rocker Switch* product line illustrates the large problem space of a typical set of products and process steps, comparable to production in CPPS work cells or assembly lines. The use case was drafted with domain experts from our industry partner, a leading systems integrator for high-performance CPPSs.

*Rocker switches* are every-day devices for controlling electrical appliances like light-switches or hairdryers. Simply put, a rocker switch consists of a socket, several contacts that can be wired to the energy sources and the electrical devices, and several rockers that can be pushed to open and close the electric circuits. Rocker switches are usually realized as switch circuit variants depending on how many functions a switch can control. The rocker switch variant determines requirements for the sequence of assembling the parts, i.e., the assembly sequence. Our industry partner plans a CPPS for manufacturing a product line of rocker switches. The CPPS should be able to produce at least 12 defined rocker switch variants, each requiring up to 60 assembly steps (with around 70% similar steps across the product variants) for the final product. The assembly sequences have several degrees of freedom concerning the order of steps (about 30% of the steps can be executed in different sequences) that define the CPPS architecture. The complexity induced by the structural and behavioral variability leads to a *large problem and configuration space* when planning the CPPS.

For planning the CPPS, the CPPS customer provides product design prototypes for each product variant. The *basic engineer* investigates particular prototype variants and explores plausible ways to assemble them and the advantages and drawbacks of these production process design options. The basic engineer then takes design decisions creating a rough design of the CPPS based on the combination of the assembly sequences (cf. Figures 1 and 2), the properties of the products, CPPS architecture options and resources, and mostly implicit knowledge and experience. Today, basic engineers at our industry partner follow a *clone-and-own* approach using spreadsheets. They create and manage the assembly sequences for each variant and list the common and variable product parts in a matrix, which is a common approach in the industry.

Figure 1 shows sections of assembly sequences of rocker switch variants as four PPR models. We elicited the required data from several documents and spreadsheets (~300 rows × 45 columns) that engineers currently use. The first step in *Variant 1* inserts an instance of *Pole 1* pole type (see the type of product directly after the name, and the instance number of the type in parentheses) into the

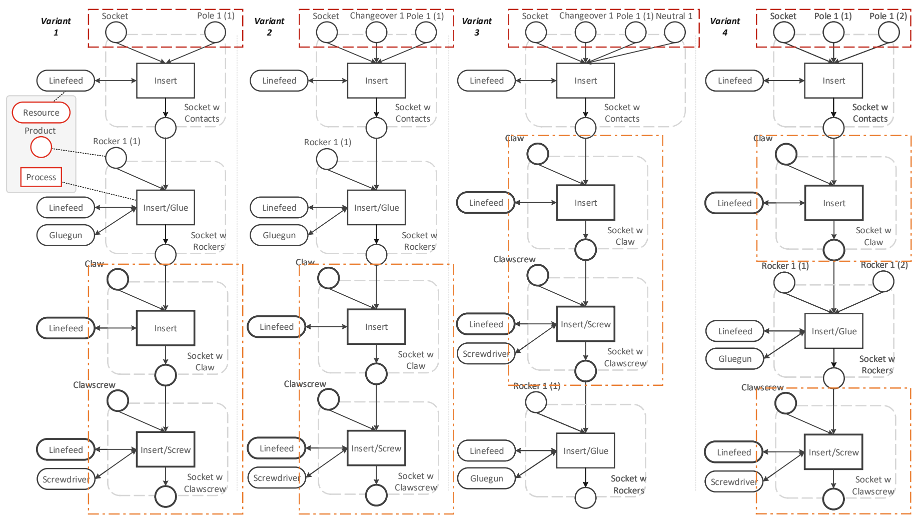


**Figure 1: Production process sections (PPR models) for four product variants in the the *Rocker Switch* product line.**

*Socket*. Then, an instance of a *Rocker 1* is inserted and glued onto the socket with the underlying pole. The final two steps place a *Claw* on the socket and screw it on with a *Clawscrew*. *Variant 2* to *Variant 4* differ in the number and types of contacts inserted and glued into the socket (see Figure 1 dashed rectangles in red) defining the *structural variability* of the products. The four PPR model instances illustrate possible assembly sequences with *behavioral variability*, as, e.g., steps for the *claw* and *clawscrew* processes can be executed in different order (see Figure 1 dot-dashed rectangles in orange).

The *process optimizer* builds on the rough CPPS plan from the basic engineer to design high-performance production process plan variants. Then, the production processes are simulated to determine their key performance indicators, such as throughput, duration, or resource consumption. Based on the rough plan and the process simulation, CPPS engineers can estimate costs as a foundation for optimizing the CPPS configuration. In addition to the PPR model shown in Figure 1, a loosely coupled process PPR model could be easier to process for the process optimizer.

Figure 2 illustrates an initial design concept of a *precedence graph* with variable product parts connected to production processes as input to production planning. The precedence graph consists of products (circles, similar to Figure 1), optional product parts (hatched pattern in orange color), dependencies between product parts (arrows pointing towards the required product part), and optional dependencies (dashed arrows). The precedence graph represents required processes (PPR model snippets) linked to product dependencies (dotted lines in red color) and requirement dependencies between processes (arrows that point towards the required process).

The precedence graph (see Figure 2) will facilitate the process optimizer in taking informed CPPS design decisions that consider the variability and dependencies of products and processes.

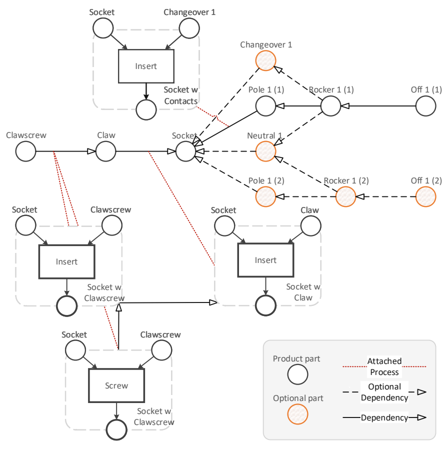
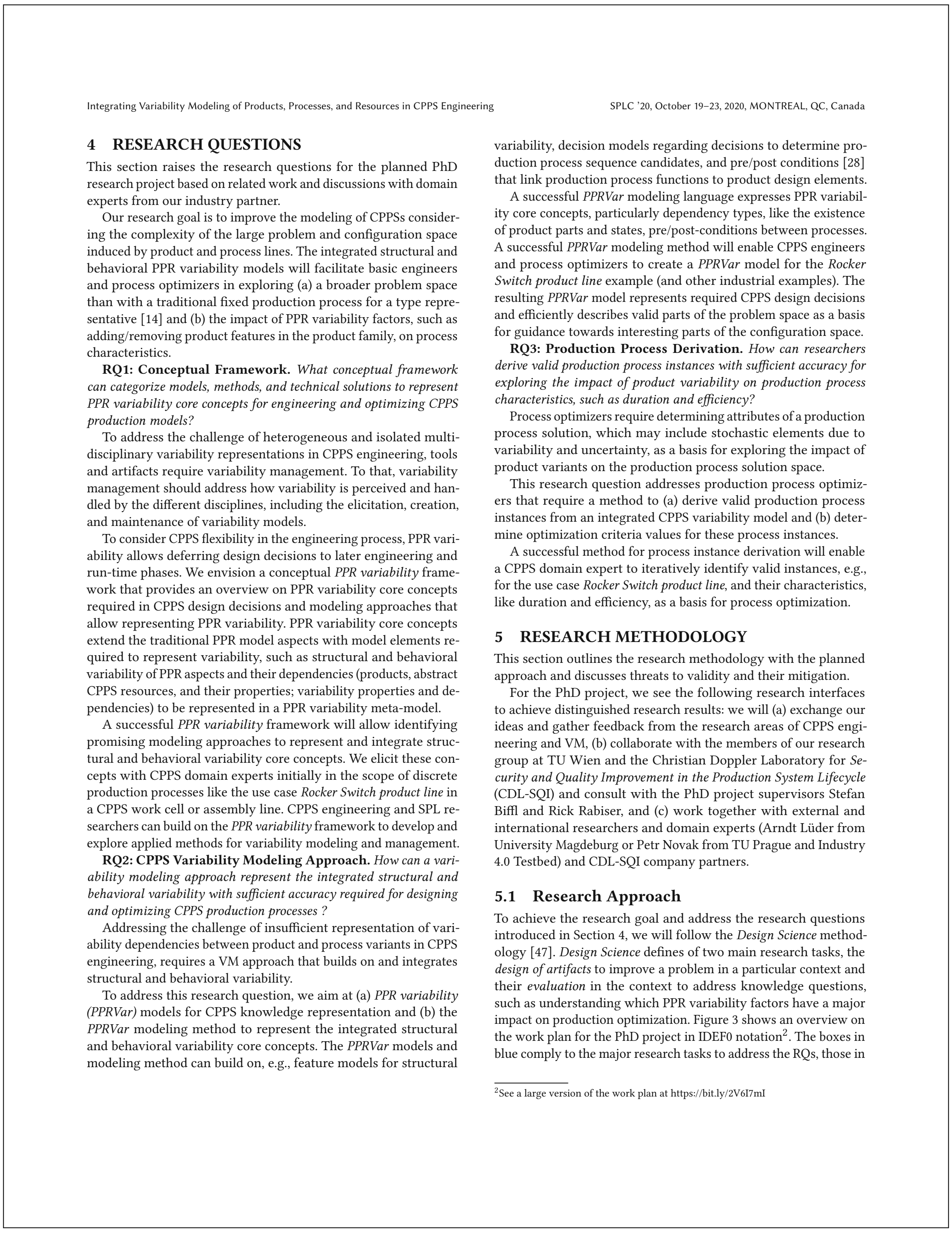


**Figure 2: Concept of a precedence graph with variability for *product* part dependencies and attached *process* snippets.**



## 4 RESEARCH QUESTIONS

This section raises the research questions for the planned PhD research project based on related work and discussions with domain experts from our industry partner.

Our research goal is to improve the modeling of CPPSs considering the complexity of the large problem and configuration space induced by product and process lines. The integrated structural and behavioral PPR variability models will facilitate basic engineers and process optimizers in exploring (a) a broader problem space than with a traditional fixed production process for a type representative [14] and (b) the impact of PPR variability factors, such as adding/removing product features in the product family, on process characteristics.

**RQ1: Conceptual Framework.** *What conceptual framework can categorize models, methods, and technical solutions to represent PPR variability core concepts for engineering and optimizing CPPS production models?*

To address the challenge of heterogeneous and isolated multi-disciplinary variability representations in CPPS engineering, tools and artifacts require variability management. To that, variability management should address how variability is perceived and handled by the different disciplines, including the elicitation, creation, and maintenance of variability models.

To consider CPPS flexibility in the engineering process, PPR variability allows deferring design decisions to later engineering and run-time phases. We envision a conceptual *PPR variability* framework that provides an overview on PPR variability core concepts required in CPPS design decisions and modeling approaches that allow representing PPR variability. PPR variability core concepts extend the traditional PPR model aspects with model elements required to represent variability, such as structural and behavioral variability of PPR aspects and their dependencies (products, abstract CPPS resources, and their properties; variability properties and dependencies) to be represented in a PPR variability meta-model.

A successful *PPR variability* framework will allow identifying promising modeling approaches to represent and integrate structural and behavioral variability core concepts. We elicit these concepts with CPPS domain experts initially in the scope of discrete production processes like the use case *Rocker Switch product line* in a CPPS work cell or assembly line. CPPS engineering and SPL researchers can build on the *PPR variability* framework to develop and explore applied methods for variability modeling and management.

**RQ2: CPPS Variability Modeling Approach.** *How can a variability modeling approach represent the integrated structural and behavioral variability with sufficient accuracy required for designing and optimizing CPPS production processes ?*

Addressing the challenge of insufficient representation of variability dependencies between product and process variants in CPPS engineering, requires a VM approach that builds on and integrates structural and behavioral variability.

To address this research question, we aim at (a) *PPR variability (PPRVar)* models for CPPS knowledge representation and (b) the *PPRVar* modeling method to represent the integrated structural and behavioral variability core concepts. The *PPRVar* models and modeling method can build on, e.g., feature models for structural variability, decision models regarding decisions to determine production process sequence candidates, and pre/post conditions [28] that link production process functions to product design elements.

A successful *PPRVar* modeling language expresses PPR variability core concepts, particularly dependency types, like the existence of product parts and states, pre/post-conditions between processes. A successful *PPRVar* modeling method will enable CPPS engineers and process optimizers to create a *PPRVar* model for the *Rocker Switch product line* example (and other industrial examples). The resulting *PPRVar* model represents required CPPS design decisions and efficiently describes valid parts of the problem space as a basis for guidance towards interesting parts of the configuration space.

**RQ3: Production Process Derivation.** *How can researchers derive valid production process instances with sufficient accuracy for exploring the impact of product variability on production process characteristics, such as duration and efficiency?*

Process optimizers require determining attributes of a production process solution, which may include stochastic elements due to variability and uncertainty, as a basis for exploring the impact of product variants on the production process solution space.

This research question addresses production process optimizers that require a method to (a) derive valid production process instances from an integrated CPPS variability model and (b) determine optimization criteria values for these process instances.

A successful method for process instance derivation will enable a CPPS domain expert to iteratively identify valid instances, e.g., for the use case *Rocker Switch product line*, and their characteristics, like duration and efficiency, as a basis for process optimization.

## 5 RESEARCH METHODOLOGY

This section outlines the research methodology with the planned approach and discusses threats to validity and their mitigation.

For the PhD project, we see the following research interfaces to achieve distinguished research results: we will (a) exchange our ideas and gather feedback from the research areas of CPPS engineering and VM, (b) collaborate with the members of our research group at TU Wien and the Christian Doppler Laboratory for *Security and Quality Improvement in the Production System Lifecycle* (CDL-SQI) and consult with the PhD project supervisors Stefan Biffl and Rick Rabiser, and (c) work together with external and international researchers and domain experts (Arndt Lüder from University Magdeburg or Petr Novak from TU Prague and Industry 4.0 Testbed) and CDL-SQI company partners.

### 5.1 Research Approach

To achieve the research goal and address the research questions introduced in Section 4, we will follow the *Design Science* methodology [47]. *Design Science* defines of two main research tasks, the *design of artifacts* to improve a problem in a particular context and their *evaluation* in the context to address knowledge questions, such as understanding which PPR variability factors have a major impact on production optimization. Figure 3 shows an overview on the work plan for the PhD project in IDEF0 notation[2]. The boxes in blue comply to the major research tasks to address the RQs, those in

[2]See a large version of the work plan at https://bit.ly/2V6I7mI

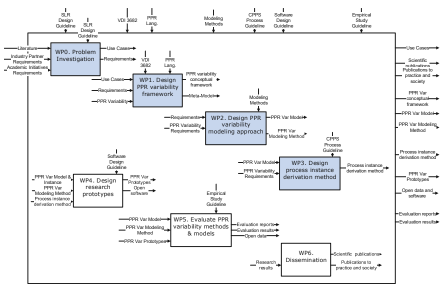

SLR Design Guideline
SLR Design Guideline
VDI 3682
PPR Lang.
Modeling Methods
CPPS Process Guideline
Software Design Guideline
Empirical Study Guideline
Literature
Industry Partner Requirements
Academic Initiatives Requirements
WP0. Problem Investigation
Use Cases
Requirements
VDI 3682
PPR Lang.
Use Cases
Requirements
PPR Variability
WP1. Design PPR variability framework
PPR variability conceptual framework
Meta-Model
Modeling Methods
Requirements
PPR Variability Requirements
WP2. Design PPR variability modeling approach
PPR Var Model
PPR Var Modeling Method
CPPS Process Guideline
PPR Var Model
PPR Variability Requirements
WP3. Design process instance derivation method
Process instance derivation method
Software Design Guideline
PPR Var Model & Instance
PPR Var Modeling Method
Process instance derivation method
WP4. Design research prototypes
PPR Var Prototypes
Open software
Empirical Study Guideline
PPR Var Model
PPR Var Modeling Method
PPR Var Prototypes
WP5. Evaluate PPR variability methods & models
Evaluation reports
Evaluation results
Open data
Research results
WP6. Dissemination
Scientific publications
Publications to practice and society
Use Cases
Scientific publications
Publications to practice and society
PPR Var conceptual framework
PPR Var Model
PPR Var Modeling Method
Process instance derivation method
PPR Var Prototypes
Open data and software
Evaluation reports
Evaluation results

**Figure 3: Research methodology and work plan.**

white comply to the iterative tasks of artifact design and evaluation and dissemination over the research process.

**WP1. Design *PPR variability* framework.** This research task will result in the *PPR variability* framework and a *PPR variability (PPRVar)* meta model on PPR variability core concepts, in particular, (a) dependencies between structural variability of products (and abstract CPPS resources) and behavioral process variability and (b) binding times for PPR variability options. The conceptual PPRVar framework will categorize models, methods, and technical solutions to represent and integrate PPR variability core concepts building on existing PPR and VM approaches. The conceptual *PPR variability* framework should serve as a blueprint that CPPS researchers can build upon to develop and explore applied methods for handling variability in the CPPS engineering lifecycle.

We will collaborate with researchers at the CDL-SQI to collect data on use cases that concern design decisions on variability and binding time, and to elicit requirements for the conceptual PPRVar framework with potential users, such as CPPS engineers and CPPS optimization researchers in academic environments, including the *TU Wien pilot factory*[3] and the *TU Prague Testbed for Industry 4.0*[4], and at industry partners. We will conduct an SLR [22] on structural and behavioral VM approaches, and approaches to integrate them, leading to the PPRVar framework categories, approaches, and their characteristics and limitations. We will evaluate the PPRVar framework in an expert survey with CPPS engineering and SPL researchers domain and experts.

**WP2. Design *PPR variability* modeling approach.** This research task will result in the *PPRVar modeling language* and *method* to represent integrated structural and behavioral PPR variability, including abstract CPPS resources. The *PPRVar* modeling language and method will build on suitable modeling approaches identified in *WP1*, such as a DSL to foster machine-readability and simple exchange, allowing easy manipulation for experts and the future use of versioning systems for storage and change detection. PPRVar models represent integrated PPR variability as a basis for CPPS engineers to define, defer, and take PPR design decisions at different points during the CPPS engineering process.

[3] http://pilotfabrik.tuwien.ac.at/en/die-pilotfabrik-der-tu-wien/themenprofil/
[4] https://www.ciirc.cvut.cz/teams-labs/testbed/

We will iteratively (a) extend models for CPPS knowledge representation considering variability requirements from research and practice towards integrating structural and behavioral CPPS variability core concepts in the PPRVar modeling language using established design guidelines and (b) design the PPRVar modeling method to guide designing PPRVar models based on the PPRVar modeling language employing proven modeling method guidelines. We will collaborate with CPPS and optimization researchers to evaluate the utility and accuracy of the resulting models in prototype evaluations and case studies on selected industrial and academic use cases, like the *Rocker Switch product line*, with requirements coming from basic engineers and process optimizers.

**WP3. Design process instance derivation method.** This research task will result in (a) a method to derive valid process instances and their properties from PPRVar models (*WP2*) and (b) selected production process instances as input to production process evaluation and optimization. Basic engineers and process optimizers can compare process instances and their properties to analyze the impact of process differences on process characteristics and to adjust the PPRVar model in case of missing or too stringent variability or dependency definitions. An interesting question is when to bind design decisions: during CPPS engineering or run time, trading off benefits from production flexibility for design complexity and risk at run time. Depending on the selected binding time of design decisions, the design of exchanged artifacts will differ and are likely to require different user and CPPS capabilities to handle variability.

We will liaise with users to define requirements for valid production process instances for a product family. We will build on approaches and frameworks, like SMT solvers [43], to design a method to derive valid production process instances for a product family and validate configurations for the integrated CPPS variability model with requirements from CPPS domain experts.

**WP4 Design research prototypes.** We will collaborate with technical experts to design and implement the technical artifacts, such as software prototypes, required to evaluate the research results. We aim at providing open data and software to the research communities as a foundation for future research.

**WP5 Evaluate PPRVar models and methods.** For the evaluation of the PPRVar models and methods, we plan to focus on our approach's utility and efficacy. We focus on these criteria as it is essential for basic engineers and process optimizers to draft a CPPS design and calculate a cost estimate with minimal effort.

Therefore, we will design and conduct case studies [40] with stakeholders from CPPS engineering and researchers. In the study on utility and efficacy, we aim to employ an existing project from our industry partner and let engineers design the variability of the CPPS with our approach. To evaluate the utility, we will assess in interviews whether the engineers were able to model the required product and process variability in the study context. We also study if our approach complements their incomplete and scattered models or even has the potential to replace some of them. To evaluate the efficacy, we will determine if the assumed gained benefit of modeling the CPPS variability with our approach outweighs the additional effort. Further, we compare our PPRVar approach with their traditional approach concerning the engineers' goal of minimal engineering effort. We plan to address further criteria in a later phase: the expressiveness and usability of our PPRVar approach.

A first case study candidate concerns modeling the use case *Rocker Switch product line* with a group of basic engineers from our industry partner, building on the GQM template [8] to define data collection goals and artifacts.

We aim at sharing evaluation data with the scientific community to validate and extend our research results.

### 5.2 Threats to Validity

We consider the following threats to validity [48]. *Limited scope of use cases* in discrete manufacturing may miss important concepts and dependencies. *Limited modeling approaches* may not fit perfectly to concepts in use cases and each other, leading to issues with integrated modeling. However, we are confident that even an initial solution will provide valuable stepping stones for researchers to improve engineering support and illustrate limitations.

*Research bias* may stem from selecting the wrong study system for evaluation, bias during requirements identification and artifact construction, and data collection and analysis. We aim at mitigating these risks utilizing collectively developed and reviewed research protocols in a team of researchers, selecting representative data and case study participants and basing our work on representative industrial case studies such as the *Rocker Switch product line.* Further, we will follow best-practice research guidelines for constructing theories, designs, and empirical evaluations.

## 6 TOWARDS INTEGRATED PPR VARIABILITY MODELING

This section presents preliminary results and a research agenda for integrating variability in CPPS engineering.

### 6.1 Preliminary results

Table 1 shows the achieved and planned publications. In Meixner et al. [30] we (a) elicited important challenges and requirements towards variability in CPPS engineering and (b) introduced a superimposed PPR model to represent CPPS variability building knowledge for *WP0/RQ1/RQ2.* In Meixner et al. [31] we presented a prototype to extract features from PPR models to support VM from exiting artifacts contributing to *WP0-WP2/RQ1/RQ2.* In Kathrein et al. [21], we extended the FPD to (a) represent *abstract* PPR aspects usable for abstract features and (b) *consistency expressions* as a basis for explicitly describing, i.a., variability dependencies contributing to *WP1/RQ2.* In Meixner et al. [28] we extended the consistency expressions to (a) link process steps to product properties via pre-/post-conditions and (b) derive test cases for testable PPR models contributing to *WP2/RQ1/RQ2.* In Meixner et al. [27] we drafted a DSL to make constraints executable building knowledge for *WP2/WP3/RQ2/RQ3.* To enable an improved basis for process optimization, we introduced the concept of modeling PPR with skills, abstract, vendor-independent descriptions of CPPS resources, and variability contributing to *WP1/WP2/RQ1/RQ2.*

| Title/Topic (Status) | Year | Venue | RQs |
|---|---|---|---|
| Production-Aware Analysis of CPPS Eng. Processes (P) [21] | 2019 | ICEIS | RQ1/RQ2 |
| Modeling Variability of PPR in CPPS Engineering (P) [31] | 2019 | SPLC WS | RQ1 |
| Feature Identification for Model Variants in CPPS (P) [30] | 2020 | VaMoS | RQ2 |
| Modeling Expert Knowledge for CPPS Resource Selection for a Product Portfolio (A) [29] | 2020 | ETFA | RQ1/RQ2 |
| Test Cases from Product and Process Model Preconditions (A) [28] | 2020 | ETFA | RQ2 |
| Towards a DSL for PPR Constraints (S) [27] | 2020 | ETFA | RQ2 |
| Requirements for managing CPPS variability (Pl) | 2021 | ICPS | RQ1 |
| Requirements for a CPPS variability framework (Pl) | 2021 | VaMoS | RQ1 |
| CPPS variability Framework (Pl) | 2021 | SPLC | RQ2 |
| SLR on structural and behavioral variability (Pl) | 2021 | IST | RQ2 |

**Table 1: Publication plan (Pl/Planned, S/Submitted, A/Accepted, P/Published work).**

### 6.2 Research Agenda

In the next 12 months, we will follow the research plan (see Figure 3). We will use an agile research approach with frequent feedback from collaborating researches and industry partners in 3-5 week sprints. Building on the preliminary research results, we work on the following work packages: (1) The *problem investigation* (*WP0*) extending it as required to gather requirements and use cases interviewing CPPS engineers and researchers as the basis for *WP1.* We will publish our results to *VaMoS 2021* and *ICPS 2021.* (2) The rough design of a *PPR variability framework* (*WP1*) concerning *RQ1.* To that, we will conduct an SLR on structural and behavioral VM, which we will publish to *IST* in 2021. From the conclusions drawn in (1) and the SLR, we will design a framework for PPR variability in CPPSs, planned for *SPLC 2021.* In parallel and based on the results from *WP0* and *WP1* we will work on *WP4* to design prototypes as a basis for evaluation and on *WP5* to design an evaluation framework based on research guidelines and fit for the particular research context.

## 7 CONCLUSION

*Cyber-Physical Production Systems* enable the flexible and optimized production of customized products on employing complex production processes. Modeling the variability of CPPS is crucial to achieve this goal but challenging due to the weak understanding (1) of variability in heterogeneous, dispersed artifacts leading to incomplete variability representations and (2) impact of changes in the product family on the CPPS design. This paper aims to motivate the need for an integrated approach to model structural and behavioral in CPPS models to support CPPS engineers and optimizers and proposes a PhD research project to conduct this research. The planned research will provide the SPL and CPPS engineering research communities with (a) models and methods on integrated CPPS variability modeling, (b) use case data from CPPS engineering, and (c) shared empirical data and evaluations from our research. The PhD project aims to provide basic engineers and production process optimizers with sufficient CPPS variability modeling methods. These should integrate structural and behavioral CPPS variability to effectively and efficiently explore the complex problem and configuration space and the impact of changes in the product features on CPPS designs.

The planned research should foster discussion on the impact of heterogeneous artifacts on VM and options to combine the variability of structure and behavior while maintaining their integrity.

We would like to ask the advisory committee: (1) How could we improve the value of the research idea and the planned results for the research community? (2) How could we improve the benefits of the research approach and plan and reduce risks? (3) Whom would you recommend considering as a research interview partner or collaborator? Thank you for your valuable time and advice.

## ACKNOWLEDGMENTS

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

## REFERENCES

[1] Sofia Ananieva, Matthias Kowal, Thomas Thüm, and Ina Schaefer. 2016. Implicit constraints in partial feature models. In *Proc. of the 7th Int. FOSD Workshop, FOSD@SPLASH 2016, Amsterdam, Netherlands, October 30, 2016.* 18–27.

[2] Sven Apel, Don Batory, Christian Kaestner, and Gunter Saake. 2013. *Feature-Oriented Software Development: Concepts and Implementation.* Springer.

[3] Rabih Bashroush, Muhammad Garba, Rick Rabiser, Iris Groher, and Goetz Botterweck. 2017. CASE Tool Support for Variability Management in Software Product Lines. *Comput. Surveys* 50, 1 (2017), 14:1–14:45.

[4] Luca Berardinelli, Alexandra Mazak, Oliver Alt, Manuel Wimmer, and Gerti Kappel. 2017. Model-driven systems engineering: Principles and application in the CPPS domain. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems.* Springer, 261–299.

[5] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wąsowski. 2013. A survey of variability modeling in industrial practice. In *Proc. of the 7th Int. Workshop on Variability Modelling of Software-intensive Systems.* ACM, 7–14.

[6] Stefan Biffl, Detlef Gerhard, and Arndt Lüder. 2017. Introduction to the Multi-Disciplinary Engineering for Cyber-Physical Production Systems. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems.* Springer, 1–24.

[7] Birte Caesar, Michael Nieke, Aljosha Köcher, Constantin Hildebrandt, Christoph Seidl, Alexander Fay, and Ina Schaefer. 2019. Context-sensitive reconfiguration of collaborative manufacturing systems. *IFAC-PapersOnLine* 52, 13 (2019), 307 – 312. 9th IFAC Conf. on Manufacturing Modelling, Management and Control.

[8] Victor R Basili-Gianluigi Caldiera and H Dieter Rombach. 1994. Goal question metric paradigm. *Encyclopedia of software engineering* 1 (1994), 528–532.

[9] Lianping Chen and M. Ali Babar. 2011. A systematic review of evaluation of variability management approaches in software product lines. *IST* 53, 4 (2011), 344–362.

[10] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wasowski. 2012. Cool Features and Tough Decisions : A Comparison of Variability Modeling Approaches. In *Proc. of the 6th Int. Workshop on Variability Modelling of Software-intensive Systems.* ACM, 173–182.

[11] Patricia Derler, Edward A Lee, and Alberto Sangiovanni Vincentelli. 2012. Modeling cyber–physical systems. *Proc. of the IEEE* 100, 1 (2012), 13–28.

[12] Matthias Galster, Danny Weyns, Dan Tofan, Bartosz Michalik, and Paris Avgeriou. 2014. Variability in Software Systems - A Systematic Literature Review. *IEEE TSE* 40, 3 (mar 2014), 282–306.

[13] Volkan Gunes, Steffen Peter, Tony Givargis, and Frank Vahid. 2014. A survey on concepts, applications, and challenges in cyber-physical systems. *KSII Transactions on Internet & Information Systems* 8, 12 (2014).

[14] Kurt W. Helbing. 2018. *Typenvertreter.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1423–1428.

[15] C. Hildebrandt, A. Scholz, A. Fay, T. Schröder, T. Hadlich, C. Diedrich, M. Dubovy, C. Eck, and R. Wiegand. 2017. Semantic modeling for collaboration and cooperation of systems in the production domain. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).* 1–8.

[16] Gerald Holl, Paul Grünbacher, and Rick Rabiser. 2012. A Systematic Review and an Expert Survey on Capabilities Supporting Multi Product Lines. *IST* 54, 8 (2012), 828–852.

[17] IEC. 2013. IEC 61131-3:2013 - Programmable controllers - Part 3: Programming languages. https://webstore.iec.ch/publication/4552

[18] IEC. 2013. IEC 62264-1:2013 - Enterprise-control system integration - Part 1: Models and terminology. https://webstore.iec.ch/publication/6675

[19] Didac Gil De La Iglesia and Danny Weyns. 2015. MAPE-K Formal Templates to Rigorously Design Behaviors for Self-Adaptive Systems. *ACM Trans. Auton. Adapt. Syst.* 10, 3, Article 15 (Sept. 2015), 31 pages. https://doi.org/10.1145/2724719

[20] Paulius Juodisius, Atrisha Sarkar, Raghava Rao Mukkamala, Michal Antkiewicz, Krzysztof Czarnecki, and Andrzej Wasowski. 2019. Clafer: Lightweight Modeling of Structure, Behaviour, and Variability. *Art Sci. Eng. Program.* 3, 1 (2019), 2.

[21] Lukas Kathrein, Arndt Lüder, Kristof Meixner, Dietmar Winkler, and Stefan Biffl. 2019. Production-Aware Analysis of Multi-disciplinary Systems Engineering Processes. In *Proc. of the 21st Int. Conf. on Enterprise Information Systems - Vol. 2: ICEIS,.* INSTICC, SciTePress, 48–60.

[22] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing systematic literature reviews in software engineering. (2007).

[23] Matthias Kowal, Sofia Ananieva, Thomas Thüm, and Ina Schaefer. 2017. Supporting the Development of Interdisciplinary Product Lines in the Manufacturing Domain. *IFAC-PapersOnLine* 50, 1 (2017), 4336–4341.

[24] Jacob Krüger, Sebastian Nielebock, Sebastian Krieter, Christian Diedrich, Thomas Leich, Gunter Saake, Sebastian Zug, and Frank Ortmeier. 2017. Beyond Software Product Lines: Variability Modeling in Cyber-Physical Systems. In *Proc. of the 21st SPLC - Vol. A.* ACM, New York, NY, USA, 237–241.

[25] Anna-Lena Lamprecht, Stefan Naujokat, and Ina Schaefer. 2013. Variability Management beyond Feature Models. *Computer* 46, 11 (2013), 48–54.

[26] Edward Lee. 2015. The past, present and future of cyber-physical systems: A focus on models. *Sensors* 15, 3 (2015), 4837–4869.

[27] Kristof Meixner, Jakob Decker, Hannes Marcher, Arndt Lüder, and Stefan Biffl. 2020. Towards a Domain-Specific Language for Product-Process-Resource Constraints. In *2020 25th IEEE ETFA, Vienna, Austria, September 8-11 2020.* IEEE.

[28] Kristof Meixner, Lukas Kathrein, Arndt Lüder, Dietmar Winkler, and Stefan Biffl. 2020. Efficient Test Case Generation from Product and Process Model Properties and Preconditions. In *2020 25th IEEE ETFA, Vienna, Austria, Sep. 8-11 2020.* IEEE.

[29] Kristof Meixner, Arndt Lüder, Jan Herzog, Hannes Röpke, and Stefan Biffl. 2020. Modeling Expert Knowledge for Optimal CPPS Resource Selection for a Product Portfolio. In *2020 25th IEEE ETFA, Vienna, Austria, September 8-11 2020.* IEEE.

[30] Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2019. Towards modeling variability of products, processes and resources in cyber-physical production systems engineering. In *SPLC (B).* ACM, 68:1–68:8.

[31] Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2020. Feature identification for engineering model variants in cyber-physical production systems engineering. In *VaMoS.* ACM, 18:1–18:5.

[32] László Monostori. 2014. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP* 17 (2014), 9–13.

[33] Klaus Pohl, Günther Böckle, and Frank van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles, and Techniques.* Springer.

[34] Mikko Raatikainen, Juha Tiihonen, and Tomi Männistö. 2019. Software product lines and variability modeling: A tertiary study. *JSS* 149 (2019), 485–510.

[35] Daniela Rabiser, Herbert Prähofer, Paul Grünbacher, Michael Petruzelka, Klaus Eder, Florian Angerer, Mario Kromoser, and Andreas Grimmer. 2018. Multi-purpose, multi-level feature modeling of large-scale industrial software systems. *Software and Systems Modeling* 17, 3 (2018), 913–938.

[36] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. 2015. Architecture and behavior modeling of cyber-physical systems with MontiArcAutomaton. *arXiv preprint arXiv:1509.04505* (2015).

[37] Dieter Rombach. 2005. Integrated software process and product lines. In *Software Process Workshop.* Springer, 83–90.

[38] Marcello La Rosa, Wil M P Van Der Aalst, Marlon Dumas, and Fredrik P Milani. 2017. Business Process Variability Modeling: A Survey. *Comput. Surveys* 50, 1 (mar 2017), 2:1–2:45.

[39] Emmanuelle Rouillé, Benoît Combemale, Olivier Barais, David Touzet, and Jean-Marc Jézéquel. 2012. Leveraging CVL to manage variability in software process lines. In *Proc. 2012 19th Asia-Pacific Software Eng. Conf.*, Vol. 1. IEEE, 148–157.

[40] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *Case study research in software engineering: Guidelines and examples.* John Wiley & Sons.

[41] Miriam Schleipen, Arndt Lüder, Olaf Sauer, Holger Flatt, and Jürgen Jasperneite. 2015. Requirements and concept for Plug-and-Work. *at-Automatisierungstechnik* 63, 10 (2015), 801–820.

[42] Jocelyn Simmonds, Daniel Perovich, María Cecilia Bastarrica, and Luis Silvestre. 2015. A megamodel for software process line modeling and evolution. In *Proc. of the 2015 ACM/IEEE 18th Int. Conf. on Model Driven Engineering Languages and Systems (MODELS).* IEEE, 406–415.

[43] Joshua Sprey, Chico Sundermann, Sebastian Krieter, Michael Nieke, Jacopo Mauro, Thomas Thüm, and Ina Schaefer. 2020. SMT-based variability analyses in FeatureIDE. In *VaMoS.* ACM, 6:1–6:9.

[44] Frank van der Linden, Klaus Schmid, and Eelco Rommes. 2007. *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering.* Springer.

[45] VDI/VDE 3682 2005. Formalised process descriptions. Beuth Verlag.

[46] Birgit Vogel-Heuser and Stefan Biffl. 2016. Cross-discipline modeling and its contribution to automation. *Automatisierungstechnik* 64, 3 (2016), 165–167.

[47] Roel Wieringa. 2014. *Design science methodology for information systems and software engineering.* Springer, Berlin [u.a.].

[48] C Wohlin, P Runeson, M Höst, M Ohlsson, B Regnell, and A Wesslén. 2000. Introduction to Experimentation in Software Engineering.

### 5.1.3 A Domain-Specific Language for PPR Modeling

#### Citation

[130] **K. Meixner**, F. Rinker, H. Marcher, J. Decker, and S. Biffl. A domain-specific language for product-process-resource modeling. In *26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021*, volume 2021-September, pages 1–8. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ETFA45728.2021.9613674

#### Aim

The publication "A Domain-Specific Language for Product-Process-Resource Modeling" [130] introduces a Domain-specific Language (DSL) for PPR modeling that serves as a foundational model for explicit engineering knowledge. The publication (i) a meta–model for modeling PPR assets with their variability and constraints, (ii) an instantiation of the model as DSL to support explicit knowledge modeling, (iii) a constraint evaluation mechanism with recursive SQL queries in PostGres, and (iv) a real-world case study of a product line in the PPR–DSL.

#### Contribution to the thesis

This publication contributes to the research goals CPPS knowledge model with variability and constraints (**G1**.), PPR–DSL constraint evaluation method (**G2**.), PPR–DSL prototype (**G4**.), case study evaluation (**G5**.).

This publication contributes to **RQ1**. and **RQ3**. by addressing the VDI 3695 measures of *models and description languages re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *variability management* C3., *enhanced quality and consistency* C4., *facilitation of interdisciplinary collaboration* C5..

#### Abstract

CPPSs are envisioned as next-generation adaptive production systems combining modern production techniques with the latest information technology. In CPPS engineering, basic planners define the functional relations between PPR views to specify valid production process and resource designs that fulfill the customer requirements. Using the Formalised Process Description (FPD) standard (VDI 3682) allows to visually model these PPR views but is hard to process by machines and insufficiently defined formally. In this paper, we present the design of a DSL, the PPR–DSL, to effectively and efficiently represent PPR aspects and evaluate constraints defined for these aspects. We illustrate the PPR–DSL with the use case rocker switch, abstracted from an industrial use case. We identify requirements and iteratively design and evaluate the PPR–DSL. We show that the PPR–DSL can model (a) the functional view of CPPSs and (b) define and efficiently evaluate constraints of a CPPS using technologies well-established in industry. We argue

that the PPR–DSL provides a valuable contribution for the community and industry to describe PPR aspects and evaluate constraints on these aspects. This way, PPR model can be defined and evaluated more easily for researchers and/or practitioners.

# A Domain-Specific Language for Product-Process-Resource Modeling

Kristof Meixner[1,2], Felix Rinker[1,2], Hannes Marcher[1,2], Jakob Decker[1,2], Stefan Biffl[2,3]
[1]Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI),
[2]Inst. of Information Systems Eng., TU Wien and [3]Ctr. Digital Production, Austria
E-Mail: [first.last]@tuwien.ac.at

2021 IEEE 26th International Conference on Emerging Technologies and Factory Automation (ETFA) | 978-1-7281-2989-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ETFA45728.2021.9613674

***Abstract*—Cyber-Physical Production Systems (CPPSs) are envisioned as next-generation adaptive production systems combining modern production techniques with the latest information technology. In CPPS engineering, basic planners define the functional relations between Product-Process-Resource (PPR) views to specify valid production process and resource designs that fulfill the customer requirements. Using the *Formalised Process Description* standard (VDI 3682) allows to visually model these PPR views but is hard to process by machines and insufficiently defined formally. In this paper, we present the design of a Domain Specific Language (DSL), the PPR DSL, to effectively and efficiently represent PPR aspects and evaluate constraints defined for these aspects. We illustrate the PPR DSL with the use case *rocker switch*, abstracted from an industrial use case. We identify requirements and iteratively design and evaluate the PPR DSL. We show that the PPR DSL can model (a) the functional view of CPPSs and (b) define and efficiently evaluate constraints of a CPPS using technologies well-established in industry. We argue that the PPR DSL provides a valuable contribution for the community and industry to describe PPR aspects and evaluate constraints on these aspects. This way, PPR model can be defined and evaluated more easily for researchers and/or practitioners.**

***Index Terms*—Digitalization, CPPS Engineering, CPPS Optimization, Model-Based Engineering, PPR Modeling.**

## I. Introduction

*Cyber-Physical Production System (CPPS)* engineering is a multi-disciplinary task that involves experts from several domains with different views on the system. These experts collaborate to achieve the common goal of planning and designing a CPPS effectively and efficiently [1], [2]. One of these domains is functional planning, where *basic planners* define the CPPS functionality in, so-called *assembly sequences*, and formulate constraints between the CPPS aspects that specify valid designs considering product requirements. However, for CPPS engineering, it is crucial to share the constraints with later engineering phases [3], [4], like detail planning and commissioning, as a foundation for verifying the correctness and efficiency of the CPPS and its behavior [5]. However, there is insufficient support for knowledge representation on the CPPS's *Product-Process-Resource (PPR)* aspects and constraints between them.

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. The competence center CDP is funded within the framework of COMET – Competence Centers for Excellent Technologies by BMVIT, BMDW, and the federal state of Vienna, managed by the FFG.

The *Formalised Process Description (FPD)* [6] for *PPR* modeling and its extensions [7], [8] provide visual representations to facilitate multi-view modeling of the functional view on the CPPS in the basic planning phase. However, recent works [6]–[8] do not define important aspects of the formal models, like the concrete syntax of the constraints and their functionality. This shortcoming makes such representations hard to process for machines impeding automated testing and verification. Model-based, machine-readable, and easily exchangeable engineering representations are the foundation for bridging gaps in the knowledge transfer [9] by providing common concepts [10] between engineering domains. *Domain Specific Languages (DSLs)* potentially reduce cost and foster portability, reliability, and testability of models [11]. DSLs are, in contrast to general purpose programming languages like C#, languages with a set of concepts and rules tailed to a particular domain. A textual DSL for the FPD and its extensions and its relevant constraints should facilitate efficient processing both for human domain experts and machines.

To tackle these shortcomings, we raise the main Research Question (RQ) on **PPR Constraint Modeling and Evaluation** (cf. Section III): *Which conceptual representation of PPR models and constraints (1) facilitates their processing by humans and machines, in particular, (2) constraint evaluation in industrial contexts?*

In previous work [12], [13], we motivated a basic design for a Product-Process-Resource Domain-Specific Language (PPR DSL) with constraints and presented a research agenda. In this paper, we build on and extend our previous work providing a full design of the PPR DSL with constraint evaluation. Following the *Design Science* approach [14], we iteratively design and evaluate the PPR DSL with domain experts from our industry partner. We apply the PPR DSL to the *Rocker Switch* use case from our industry partner to demonstrate the PPR DSL's feasibility.

We summarize related work in Section II, motivate detailed RQs in Section III, and introduce the *Rocker Switch* use case in IV. In Section V, we introduce the PPR DSL approach addressing the RQs. We demonstrate the feasibility and applicability of our DSL approach in Section VI, followed by a discussion in Section VII. Finally, Section VIII concludes and gives an outlook on future work.

## II. Related Work

This section summarizes related work on CPPS engineering, PPR concepts and their constraints, and DSLs.

**Cyber-Physical Production System Engineering.** CPPS engineering processes are often customized for *intra-disciplinary* activities [7]. Therefore, shortcomings regarding the definition of common concepts [10], discipline-specific tools, and proprietary and heterogeneous artifacts limit the effectiveness and efficiency of *interdisciplinary* knowledge exchange. For example, spreadsheets require expert interpretation due to varying and implicit semantics. However, the CPPS engineering disciplines and their concepts are linked, requiring comprehensive models to express the requirements of products toward designing a CPPS [9].

Schleipen *et al.* [15] coined the term *PPR* based on the three aspects *product* with its properties, *process* that produces a product, and *resource* that executes production processes. The FPD [6] represents the PPR aspects in a technology- and tool-agnostic way by defining a graphical notation and a data model for the functional view on a CPPS. In this paper, we build on these PPR foundations.

**Product-Process-Resource Concepts and Constraints.** The *Formalised Process Description (FPD)* [6] enables the definition of production sequences by specifying relations between input and output products and production processes. However, the FPD does not provide means to formulate consistency constraints, such as the applicable torque for a screw or the maximal weight of a set of CPPS resources. Such constraints are crucial in CPPS engineering to formulate requirements, e.g., from products, that must not be violated to yield the necessary product quality or might even break the CPPS if violated. Kathrein *et al.* [7] extended the FPD with constraint expressions and with abstract resources that act as placeholders to be specified later, and with consistency constraints that define dependencies between the PPR aspects. In this paper, we build on the FPD and its extensions [7] to investigate how to efficiently define and evaluate PPR constraints with technology that is well-established in industry.

**Domain-Specific Languages on Constraints.** A DSL is a language to efficiently describe a specific kind of problem in a domain using domain-specific concepts, such as PPR. In contrast, a general-purpose language provides only general domain-agnostic concepts [16] that require adaptation to a domain. The are several approaches recommended for designing and creating the structure of a DSL [11], [17]. The Meta Object Facility (MOF)[1] is a standard for model-driven engineering, defining an architecture for the classification of models. A popular approach with tool support is the *Eclipse Modeling Framework (EMF)* [18]. However, the technical footprint of the EMF is quite large and EMF is hard to include in applications other than the *Eclipse IDE*[2].

Constraint definition and evaluation require a specific syntax and engine capable of specifying and evaluating the constraints. The *Object Constraint Language (OCL)* is a framework standardized by the *Object Management Group* [19]. OCL is quite expressive, but may be difficult to use for CPPS engineers, who are not software modeling experts. An alternative is the *Structured Query Language* (SQL) [20], a goal-oriented query language for relational databases to formulate and evaluate constraints on databases. However, it is not clear how to leverage SQL capabilities for evaluating constraints on PPR aspects. In this paper, we explore how to efficiently evaluate PPR constraints building on SQL capabilities, which are well-known and accessible to CPPS engineers.

The Reference Architecture Model Industry 4.0 (RAMI 4.0) is a 3-dimensional model that describes a technical asset in terms of roles, its lifecycle and value stream, and allows the classification into various hierarchy levels, each on a different axis [21]. The proposed PPR DSL assists the development phase of a station component and allows a consistent description from a functional viewpoint. Therefore, the contributions of this paper fit into the functional layer, the development phase of lifecycle and value stream, and the hierarchy level of a station component.

## III. Research Questions

CPPS engineers require approaches beyond visual representations of PPR models to model their engineering knowledge. We aim to define a DSL for the PPR approach that can be used efficiently to formalize PPR models. Therefore, the following research questions guide our research for a PPR DSL.

***RQ1. PPR Constraint Modeling.*** *Which domain model and textual DSL can represent PPR models and constraints for efficient processing by humans and machines?*

We identify requirements for a set of elements of a PPR DSL to support domain experts to engineer CPPSs. To address these requirements, we design (a) a meta-model for a PPR DSL and (b) the PPR DSL.

***RQ2. PPR Constraint Evaluation.*** *How can PPR constraints, formulated in a DSL, be evaluated with a technology that is established in the production system industry, such as relational databases?*

For CPPS engineers it is important to formulate consistency constraints during the design time of the CPPS to validate design decisions across the involved disciplines. However, constraint evaluation frameworks from model-driven software engineering, like OCL, are often hard to integrate to existing engineering tool landscapes and the run-time context. Therefore, we investigate how to map the concepts of PPR models with constraints to a well-established industrial technology, such as relational database technology, for efficiently evaluating the consistency constraints.

In workshops with several domain experts at a company partner in discrete manufacturing, we elicited the following requirements towards a PPR DSL with constraints, and technologies for their implementation.

**R1. PPR representation.** The DSL shall represent (R1.1) the *PPR aspects* – products, processes, and resources and their *attributes*. (R1.2) the *relationships* between the PPR aspects

[1]MOF: https://www.omg.org/mof
[2]Eclipse IDE: https://www.eclipse.org/ide/

conforming to the VDI 3682 standard [6], including *is-part* relations, to model PPR sub-structures.

**R2. Reusable templates for PPR aspect definitions.** To foster the reuse of definitions, the PPR DSL shall represent *abstract PPR aspects*. For instance, a resource might be defined in an abstract way with high-level attributes as a foundation for defining concrete resources that inherit these attributes.

**R3. Constraint representation.** The DSL shall represent constraints between the PPR aspects, including the *aggregation* over multiple levels (*is-part* relations) and attributes (*sum* or *average* function).

**R4. Focus on domain-specific concepts.** The DSL shall focus on domain-specific concepts, such as PPR concepts, to facilitate the DSL use with engineers in various disciplines, typically with limited education in software engineering.

**R5. Technology-agnostic DSL definition and evaluation.** The DSL shall be technology agnostic to support the integration with typical industrial standard software environments and contexts, e.g., for mapping DSL concepts to query languages that are well-known and accessible in the target domain and can express/execute recursive queries, e.g., SQL, OCL, Cypher, or DataLog.

To address the RQs and these requirements, we iteratively design and develop our approach for a PPR DSL with constraint evaluation following the *design science* approach [14]. We evaluate our results in a feasibility study utilizing the illustrative use case *Rocker Switch* (cf. Section IV).

## IV. Illustrative Use Case *Rocker Switch*

This section presents the use case *Rocker Switch* as running example. The use case stems from an analysis of a CPPS design from a cooperation with an industry partner, showing an abstracted design to honor intellectual property rights.

Rocker switches are everyday appliances to control devices, like lights or sun-blinds. The bottom part of Figure 1 shows a schematic view of the core of a rocker switch. Basically, this core consists of a *socket* (in grey), several contacts, like *pole* (in orange), *neutral*, or *off*, that are held in place by *screws* (in pink), and a number of *rockers* (in green). An analysis of the rocker switch's CPPS design revealed that the system manufactures 12 different rocker switch variants building a product family [22]. A rocker switch variant can require up to sixty individual production steps, resulting in more than 600 assembly sequence parts that engineers need to design and maintain.

Figure 1 shows a part of the assembly sequence of a rocker switch as a PPR model with an extended view on the CPPS resources. In the first step, *Work Cell 1* inserts the *Pole* contact into the *Socket* in an *Insert/Press* process. In the second step, *Work Cell 2* inserts the *Rocker* into the *Socket*. In the third step, *Work Cell 3* inserts and screws *Screw 1* and *Screw 2* into the socket. Therefore, *Work Cell 3* has two *Screwdrivers* each consisting of a *Bit* and a *Drive*. Each *Drive* has a *Transformer* that is controlled by a *Screwing Controller*. The *Screwing Controllers* are lead by *Robot Controllers* that are orchestrated by an *Industrial PC (IPC)*.

For the design of the use case, we identified two important constraints. Constraint *C1* defines the screwdrivers not to exceed the maximum torque specified by the designs of the screws and the socket. *C1* is relevant both at design time and at run time. Constraint *C2* states that all resources connected to the screwdriver must not exceed a power consumption of 50 Watt.

Today, domain experts often create and manage assembly sequences in spreadsheets that lack semantic relationships, an approach that is time-consuming and error-prone. Further, constraints like *C1* and *C2* are very hard to formulate and evaluate in tools like spreadsheets, as the required information and structural data are often scattered across several heterogeneous documents. Therefore, a PPR DSL providing suitable capabilities with tool support to model these assembly sequences could reduce effort and risk in CPPS engineering.

## V. Domain-Specific PPR Language

This section introduces the design of the PPR DSL, illustrated with examples from the use case *Rocker Switch*.

### *A. Domain-Specific Product-Process-Resource Model*

We build on the meta-models of the VDI 3682 [6] and our previous work [8], [12] to define the domain model for the PPR DSL. The domain model of a DSL defines the concepts of the DSL and their relationships. Figure 2 shows the domain model for the PPR DSL in UML notation representing Layer 1 of the MOF. We explain the core concepts of the DSL using the *Rocker Switch* use case.

The root concept is the `AssemblySequence`, i.e., the functional production sequence that engineers want to model.

To address requirement R1.1, `AssemblySequence` contains the `Product`, `Process`, and `Resource` concepts,[3] for instance, the product *Screw*, the process *Insert/Screw*, and the resource *IPC* in the use case (cf. Figure 1). Instances of these three concepts can be declared abstract with the `isAbstract` property. This property indicates a particular concept to be a template for concrete instances, addressing requirement R2.

The PPR concepts have four reflexive relationships to address requirement R1.2. The relation `implements` indicates a concept to implement an abstract concept of similar type, inheriting its characteristics. For example, an *electric screwdriver* could implement an abstract resource *screwdriver*. The relations `parents` and `children` allow defining the containment relations *has-part* and *is-part*. For instance, the *screwdriver* has the parts *bit* and *drive*. Both concepts are required for defining multiple parents and children, e.g., to model multiple views on the system. The relation `requires` defines required other concepts, other than a containment relation. For instance, the *screwer driver* requires the *robot controller*. Similarly, the relation `excludes` defines concepts that cannot be combined with the particular aspect.

[3]Due to space limitations, Figure 2 shows the properties and reflexive relations, applicable to each of the three PPR concepts, only for the *Resource*.

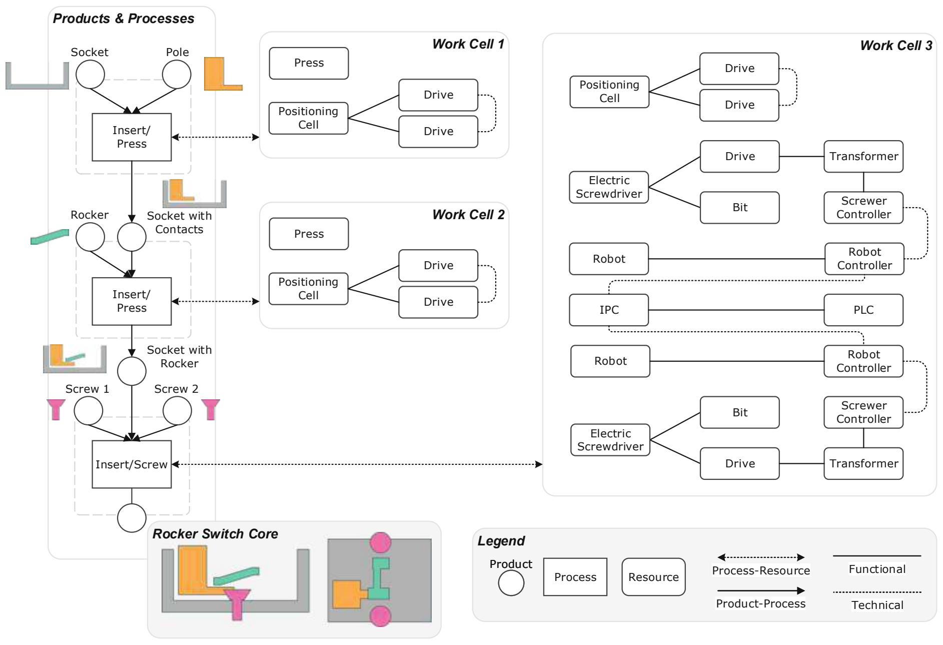


Fig. 1. Product-Process-Resource Model with constraints for a *Rocker Switch*.

To address requirement R1.2, we defined three relationships between the PPR concepts. Input products are connected to a process with `InputProductProcessRelation`, e.g., the *Socket* is connected to the process *Insert/Press* (cf. Figure 1). Output products are connected to a process with `OutputProductProcessRelation`, e.g., the output product *Socket with Contacts* is connected to the process *Insert/Press*. To relate a resource with a process, the domain model defines `ResourceProcessRelation`, e.g., the resource *IPC* is connected to the process *Insert/Screw*.

To address requirement R1.1, the `AssemblySequence` contains the `Characteristic` concept, conforming to VDI 3682 [6]. Characteristics model properties of PPR concepts, e.g., the *torque* that a screw can take, and of relationships. Characteristics are globally defined to allow their reuse in the model, addressing requirement R2.

The `AssemblySequence` also holds the `Constraint` concept allowing to include PPR aspects and global attributes. We describe constraints in detail in Section V-C.

### B. Domain-Specific Product-Process-Resource Language

To design the textual representation of the PPR DSL requires mapping the domain model to language constructs, like the keywords, utilizing an iterative DSL design approach [17]. We aimed at one-to-one mappings between domain model concepts and language constructs and at introducing keywords that facilitate understandability for CPPS engineers, addressing requirement R4.

Listing 1 shows an excerpt[4] of the PPR DSL model for the use case *Rocker Switch*. Basically, each concept is defined using a *keyword*, followed by an *ID* and a *body* in curly braces with further definitions.

The concept *Characteristic* is mapped to the keyword `Attribute`. For instance, Line 1 globally defines the attribute *MaxAllowedForce* with the torque unit *Nm*.

Lines 5 to 8 define four products. For instance, Line 7 defines product *Screw 1* with ID *S1* using the globally defined attribute *MaxAllowedForce*.

[4]A complete DSL representation of the use case *Rocker Switch* and an overview graphic of the improved PPR-design approach can be found at https://github.com/tuw-qse/ppr-dsl-case-study

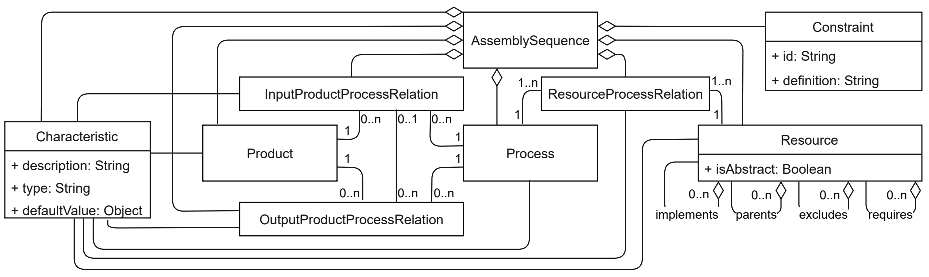


Fig. 2. The domain model of the PPR DSL with the main concepts *Product-Process-Resource* and *Constraint* (in UML notation).

The following lines define the resource *Work Cell 3*. The construct `children` defines the parts of the work cell, such as the *IPC*. Abstract concepts are defined by the keyword `isAbstract` in their body. For instance, Line 16 defines the abstract resource *Screwdriver* that is implemented in Line 21 as the concrete resource *Screwdriver 1*. The keyword `requires` allows creating logical categories or groups. Line 30 indicates that the *ScrewingController* `requires` a *RobotController*. Similarly, the keyword `excludes` can be used to model forbidden combinations, e.g., two product parts that must not be used together.

Processes can be defined using the keyword `Process`. A *InputProductProcessRelation* corresponds to the DSL keyword `inputs` in the `Process` (cf. *InsertScrew* in Listing 1) followed by a list of identifiers of the input products. Similarly, an *OutputProductProcessRelation* is defined using the DSL keyword `outputs` with corresponding identifiers. A *ResourceProcessRelation* is indicated with the keyword `resources` within the `Process` linked with identifiers of the resources. For instance, process *InsertScrew* uses resource *Work Cell 3* to execute the production step.

### *C. Constraint Definition and Mapping*

*PPR Constraint Definition.* To address requirements *R3* and *R4* and to provide an easy-to-use constraint language for CPPS engineers, we defined a simple constraint syntax. The specification of PPR constraints is divided into a *left-hand side (LHS)*, specifying the involved PPR aspects, and a *right-hand side (RHS)*, specifying the constraint.

Listing 1 (Lines 41 to 46) illustrates two constraints – *C1* and *C2*. *C1* defines the force, applied by *Screwdriver1* and *Screwdriver2* to the product *SocketWithRocker* shall not exceed the product's defined force threshold. *C2* defines all components belonging to *Screwdriver1* together shall not consume more than 50 Watt. A constraint may concern an *a-priori* unknown number of PPR aspects. For constraint *C2*, the affected PPR aspects are *Bit*, *Drive*, *Transformer*, and *ScrewerController*.

The specification of PPR aspects in a constraint can either be *explicit* or *implicit*. Constraint *C1 explicitly* specifies the involved PPR aspects: *Screwdriver1*, *Screwdriver2*, and *SocketWithRocker*. Constraint *C2 implicitly* specifies the affected PPR aspects: all components belonging to *Screwdriver1*. The implicit specification of PPR aspects can use the keywords `subtype` (for *implements* relations) and `descendants` (for *containment* relations). Both keywords are followed by a qualifier name, as a reference in the constraint text. Constraint *C2* uses the qualifier name `all`.

The RHS of the PPR constraints specifies the constraint formula. Constraints can be formulated either as an aggregation function, followed by a comparison operator and a numerical expression (cf. *C1*) or an arrow symbol, followed by a Boolean expression (cf. *C2*). Table I summarizes the supported aggregation functions and operators.

TABLE I
OPERATORS FOR PPR CONSTRAINT DEFINITION AND EVALUATION.

| Symbol | Description |
|---|---|
| `->` | For all objects on the LHS of `->`, execute the evaluation formula on the RHS. |
| `=, <, >, =>, <=` | Comparison operators |
| sum, count, avg, min, max | Aggregation operators |
| descendants\|subtype <qualifier> | Access to child/derived aspects regarding the aspect declared. Retrieving attribute values from child/derived aspects, requires a qualifier. |

*PPR Constraint Mapping for Evaluation.* The model-driven engineering community offers frameworks to implement DSLs and the evaluation of constraints (cf. Section II). However, in the industrial context it is important to use established technologies that work on the infrastructure of a production system. The *Structured Query Language (SQL)* provides standardized capabilities to query relational data models and is an established technology in industry.

To address *RQ2*, i.e., to evaluate the constraints in a well-established technology frequently used in production systems, we designed the following approach: (a) map the PPR DSL constraint syntax to intermediate representations, i.e., SQL

```
Attribute "MaxAllowedForce": {unit: "Nm"}
Attribute "AppliedForce": {unit: "Nm"}
Attribute "PowerConsumption": {unit: "W"}

Product "SwR": {name: "Socket with Rockers"}
Product "SwS": {name: "Socket with Screws"}
Product "S1": {name: "Screw 1", MaxAllowedForce: 4}
Product "S2": {name: "Screw 2", MaxAllowedForce: 4}

Resource "WC3": {name: "Work Cell 3",
  children: ["IPC", "Robot1", "SDriver1",
             "Robot2", "SDriver2", "PCell3"]}

# Definition PCell3, Drive5, Drive6, IPC, PLC, ...

Resource "ScrewDriver": { name: "Screwdriver",
  isAbstract: true, AppliedForce: 3}

Resource "SDriver1": {name: "Screwdriver 1",
  children: ["Bit1", "Drive7"],
  implements: ["Screwdriver"]}
Resource "Bit1": {name: "Bit 1"}
Resource "Drive1": {name: "Drive 7",
  PowerConsumption: 10,
  children: ["Transformer1"]}
Resource "Transformer1": {name: "Transformer 1",
  children: ["ScrewerController1"]}
Resource "ScrewerController1": {
  name: "Screwer Controller 1",
  requires: ["RController1"]}

# Definition SDriver2, ...

Process "InsertScrew": { name: "Insert/Screw",
  inputs: [ {productId: "S1"}, {productId: "S2"},
    {productId: "SwR", comesFrom: "IPOut"} ],
  outputs: [ {SwSOut: {productId: "SwS"}} ],
  resources: [ {resourceId: "WC3", minCost: 0.0} ]
}

Constraint "C1":{
    definition: "SDriver1, SDriver2
      max(AppliedForce) <= SwR.MaxAllowedForce"
}
Constraint "C2":{
    definition: "SDriver1 descendants all ->
      all.PowerConsumption < 50"
}
```

Listing 1: PPR DSL section of the *Rocker Switch* use case (cf. full example at https://github.com/tuw-qse/ppr-dsl-case-study)

```
1  WITH combined AS (
2   SELECT * FROM resourceAttributeView
3   WHERE id='Screwdriver1' AND
4    attributeId='AppliedForce'
5   UNION SELECT * FROM resourceAttributeView
6   WHERE id='Screwdriver2' AND
7    attributeId='AppliedForce')
8
9  SELECT * FROM
10  (SELECT MAX(attributeValue) AS aggregation
11   FROM combined) AS aggr,
12  productAttributeView SwR0
13 WHERE SwR0.id='SwR' AND
14  SwR0.attributeId='MaxAllowedForce' AND
15  NOT (aggr.aggregation <= SwR0.attributeValue)
```

Listing 2: SQL Query implementing Constraint *C1*.

queries, (b) execute the SQL query on a suitable relational database system, and (c) observe whether a constraint violation occurred. To determine a suitable intermediate representation, we investigated the capabilities of SQL regarding constraint evaluation and designed SQL query templates with instantiated examples shown in Listings 2 and 3.

To evaluate the SQL expressions, we designed a relational model consisting of tables for the concepts *Product*, *Process*, *Resource*, and *Attribute*. The PPR tables are connected with the *Attribute* table by the following three views to provide easier access: *ProductAttributeView*, *ProcessAttributeView* and *ResourceAttributeView* respectively.

Listing 2 illustrates the result of mapping of constraint *C1* to SQL. First, the *common table expression* in Listing 2 introduces a new temporary relation *combined*. The relation *combined* encapsulates the PPR aspects, specified at the LHS of constraint *C1*. Lines 2 to 4 select the first involved element, *Screwdriver1*, with the required attribute, *AppliedForce*. Lines 5 to 7 select the second element, *Screwdriver2*, with the same attribute. The expression starting at Line 9 evaluates the constraint formula, i.e., it computes the aggregation function over the previously generated relation *combined* and joins additional relations as required. Lines 10 to 11 compute the aggregation function as a sub-query, using the SQL aggregation function *MAX* (or a similar SQL function as specified; cf. to operators in Table I). Lines 12 to 14 join additional relations required to evaluate the constraint (*SwR* with attribute *MaxAllowedForce* in the case of *C1*). Line 15 evaluates the actual constraint (RHS of *C1*).

Listing 3 illustrates the mapping of constraint *C2* to SQL. The general concept is similar to Listing 2. The first expression in Listing 3 builds a relation over the LHS of the constraint assigning the name *combined*. However, this time the first expression is slightly more complicated. As the number of involved PPR aspects (all components of *Screwdriver1*) is not known in advance, a *recursive common table expression* is necessary. Lines 2 to 4 select the root node (*Screwdriver1*). Lines 5 to 10 sequentially add all children of the root node. Table *resourceContainsResource* specifies the containment relations, i.e., `parents` or `children`. For constraint *C2*, *combined* contains *Screwdriver1* with its attributes and all children with their attributes. The second expression, starting at Line 12, computes the constraint formula (RHS of *C2*). Lines 13-14 exclude the root node (*Screwdriver1*) and restricts the attributes to *PowerConsumption*. Line 15 evaluates the constraint formula, checking whether all remaining elements have an attribute value less than 50.

```
WITH RECURSIVE combined AS
 (SELECT resourceAttributeView.*
   FROM resourceAttributeView
   WHERE id='Screwdriver1'
UNION SELECT main.*
 FROM combined,
   resourceAttributeView main,
   resourceContainsResource containment
 WHERE combined.id=containment.parentId AND
   main.id = containment.childId)

SELECT * FROM combined
WHERE combined.id != 'Screwdriver1' AND
   combined.attributeid = 'PowerConsumption' AND
   NOT (combined.attributevalue < 50)
```

Listing 3: SQL Query implementing Constraint *C2*.

## VI. Evaluation

To evaluate our approach we conducted a feasibility study. We developed a prototype of the PPR DSL and the constraint evaluation and evaluated it using the *Rocker Switch* use case. We further interviewed the domain experts of our industry partner to gather feedback on our approach.

We decided to use Java and *PostgreSQL*[5] as relational database, which provide the necessary capabilities, like recursive queries and views, for the prototype. We created parsers that read the DSL into the Java implementation of the domain model (cf. Figure 2). The prototype then persists the PPR concepts and attributes to the PostgreSQL database. The evaluation engine reads the constraint specifications, produces equivalent SQL representations and executes them against a database to determine whether a given constraint is violated. After executing the queries, the result is either empty or not. In the first case, the evaluation engine concludes that no violation is present. In the second case, the evaluation engine uses the resulting rows to determine the cause for the violation.

To investigate the performance of the approach, we measured the execution time for constraint evaluation. We used the *Java Microbenchmark Harness* framework[6] to measure the average execution time of the evaluation from the prototype[7]. To be able to compare the execution time and calculate an average execution time configured JMH to use 100 iterations. The evaluation required 8.2 milliseconds (ms) with a standard deviation of 0.45 ms for constraint *C1* and 9.3 ms with a standard deviation of 0.6 ms for constraint *C2*.

We further evaluated the PPR DSL in the industrial context. Therefore, we conducted a workshop and interviews[8] with two CPPS domain experts from our industry partner on the perceived usefulness of the PPR DSL. First, we introduced the syntax and semantics to the study participants and verified their understanding of the PPR DSL concepts. Then, the domain experts had to define a PPR DSL model from a functional model of a CPPS. Compared to their traditional approach to draw the PPR models, they found the PPR DSL easier to create and maintain. However, they also mentioned the PPR DSL syntax to be quite verbose and found ways for simplification. Compared to their traditional approach to formulate the constraints in natural language, they found the structured constraint definition and the automated evaluation useful for their engineering tasks. They also approved the implementation to software that they already use in their production systems as this approach lowers the entry barrier to using explicitly defined constraints.

[5]PostgreSQL: https://www.postgresql.org

[6]JMH framework - https://openjdk.java.net/projects/code-tools/jmh

[7]These measures include the time that Java needs to set up the queries to the PostgreSQL database.

[8]The detailed guidelines for the interviews can be found in [13].

## VII. Discussion

This section discusses our results and the research questions.

We first raised the question, which domain model and textual DSL can represent PPR models and constraints. To address RQ1, in Section V we identified requirements for a set of elements of a PPR DSL to support domain experts to engineer CPPSs. To address these requirements, we developed (a) a domain model for a PPR DSL based on the VDI 36282 standard and our previous work and (b) a language design for the PPR DSL. This goes beyond the state of the art, as most approaches only visually support the VDI 3682 standard and do not support important extensions such as abstract PPR aspects.

Secondly, we asked how PPR constraints can be evaluated with a technology that is established in the production system industry, such as relational databases? Existing DSL workbenches, such as Meta Programming System (MPS) or Eclipse Modeling Framework (EMF) [13], mostly have a high complexity and steep learning curve. Furthermore, these technologies and especially the constraint evaluation are hard to integrate into the current software landscape of many industrial production systems. To address RQ2, in Section V-C we developed a mapping of our PPR DSL constraint syntax to the SQL standard. Using these mappings the constraints can easily be evaluate on relation database systems. These database systems are well-established in the industry and frequently used in production systems to hold the production systems' data and control its functions via software. This way it is possible to easily integrate our constraint evaluation mechanism to the engineering lifecycle but also use it during runtime to validate the systems behavior.

An evaluation with domain experts from an industry partner of a prototype for our approach using Java and PostgresSQL showed that (i) our approach is reasonably efficient and (ii) that domain experts found our approach useful in the context of engineering their CPPS using the PPR DSL to share knowledge over the engineering phases and validate potential functional designs.

These research results build on the VDI 3682 standard and its extensions [12] and go beyond the state of the art in production systems engineering by providing (i) a machine-readable and easy-to-maintain formal definition of PPR models; (ii) technology-agnostic constraint definition that can be mapped to several constraint evaluation technologies; and (iii) a mapping of the technology-agnostic constraint definition for constraint evaluation in SQL, an established industrial technology.

**Threats to Validity.** *Expressiveness of constraints.* The focus of the PPR DSL constraints is on the expression of constraints on PPR aspect attributes. Therefore, the PPR DSL cannot express, e.g., whether certain aspects are missing in the design of the CPPS. However, even the current PPR DSL capabilities to express constraints seem useful for typical practitioners. *Limited Usecases.* We evaluated the PPR DSL with one use case and in a specific production system domain. However, the research design carefully selected a use case derived from a real-world industrial project that is likely to identify requirements that can be expected to be useful to some extent for similar use cases in production systems engineering.

## VIII. Conclusion

In Cyber-Physical Production System (CPPS) engineering, basic engineers want to design the functional view on a particular CPPS as PPR models based on VDI 3682 and share constraints regarding valid CPPS designs. These constraints are crucial to validate the CPPS design through the engineering phases. However, there is only limited support for representing PPR aspects and constraints. Furthermore, current approaches for constraint evaluation are often hard to integrate into the software landscape of a production system.

In this paper, we identified five requirements for a DSL to representing PPR models. We then introduced a domain model and mapped a language design for the PPR DSL. The PPR DSL allows a *representation of PPR aspects* and a formulation of *constraints between the aspects*.

One research question was how to integrate the constraint evaluation with well-established technologies in CPPS engineering. We developed a mapping of the PPR DSL constraint syntax to SQL queries to execute them on relational database systems, a well-established technology in industry.

To evaluate our approach, we developed a prototype to define and evaluate the constraints in a *PostgresSQL* database. We tested our approach in a feasibility study with the industrial use case *Rocker Switch*. The evaluation found the mapping to *PostgresSQL* to work well with reasonable duration for constraint evaluation, even on a laptop computer. Furthermore, domain experts interviewed on the perceived usability of the PPR DSL stated the DSL to be easily usable in their context and noted suggestions for simplifying some language aspects.

Overall, we were able (i) to establish a DSL to define PPR models that are based on VDI 3682 and (ii) to successfully evaluate constraints defined in the PPR DSL using SQL, a well-established technology in industry.

**Future Work.** We plan to design improved tool support for the PPR DSL including an editor with instant feedback. Feedback from practitioners provided input on simplifying the PPR DSL to improve support for engineers with limited programming background. Further, we plan to benchmark the evaluation of the constraints with different databases, including emerging technologies in industry, such as graph databases.

## References

[1] S. Biffl, D. Gerhard, and A. Lüder, "Introduction to the multi-disciplinary engineering for cyber-physical production systems," in *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 2017, pp. 1–24.

[2] L. Monostori, "Cyber-physical Production Systems: Roots, Expectations and R&D Challenges," *Procedia CIRP*, vol. 17, pp. 9–13, 2014.

[3] A. Egyed, K. Zeman, P. Hehenberger, A. Demuth, L. C. Zimmermann, and R. Kretschmer, "Maintaining consistency across engineering artifacts," in *SE*, ser. LNI, vol. P-300. Gesellschaft für Informatik e.V., 2020, p. 129.

[4] A. Lüder, J.-L. Pauly, and M. Wimmer, "Modelling consistency rules within production system engineering," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2018, pp. 664–667.

[5] M. Seitz and B. Vogel-Heuser, "Challenges for the digital transformation of development processes in engineering," in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 2020, pp. 4345–4350.

[6] "VDI/VDE 3682: Formalised process descriptions." Beuth Verlag, VDI/VDE, 2005.

[7] L. Kathrein, A. Lüder, K. Meixner, D. Winkler, and S. Biffl, "Production-aware analysis of multi-disciplinary systems engineering processes," in *Proc. of the 21st Int. Conf. on Enterprise Information Systems - Vol. 2: ICEIS,*, INSTICC. SciTePress, 2019, pp. 48–60.

[8] L. Kathrein, K. Meixner, D. Winkler, A. Lüder, and S. Biffl, "A meta-model for representing consistency as extension to the formal process description," in *ETFA*. IEEE, 2019, pp. 1653–1656.

[9] L. Berardinelli, A. Mazak, O. Alt, M. Wimmer, and G. Kappel, "Model-driven systems engineering: Principles and application in the CPPS domain," in *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 2017, pp. 261–299.

[10] F. Rinker, L. Waltersdorfer, K. Meixner, and S. Biffl, "Towards Support of Global Views on Common Concepts employing Local Views," in *ETFA*. IEEE, 2019, pp. 1686–1689.

[11] A. Van Deursen and P. Klint, "Domain-specific language design requires feature descriptions," *Journal of computing and information technology*, vol. 10, no. 1, pp. 1–17, 2002.

[12] K. Meixner, J. Decker, H. Marcher, A. Lüder, and S. Biffl, "Towards a domain-specific language for product-process-resource constraints," in *ETFA*. IEEE, 2020, pp. 1405–1408.

[13] J. Decker and H. Marcher, "A Domain-Specific Language for Connecting Product-Process-Resource Models with Dependencies," CDL-SQI, Institute for Information Systems Engineering, TU Wien, Technical Report CDL-SQI 2020-06 CDL-SQI-2020-06, Nov. 2020.

[14] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *Design Science in IS Research MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[15] M. Schleipen, A. Lüder, O. Sauer, H. Flatt, and J. Jasperneite, "Requirements and concept for plug-and-work," *at-Automatisierungstechnik*, vol. 63, no. 10, pp. 801–820, 2015.

[16] M. Fowler, *Domain-specific languages*. Pearson Education, 2010.

[17] M. van Amstel, M. van den Brand, and L. Engelen, "An exercise in iterative domain-specific language design," in *Proceedings of the joint ERCIM workshop on software evolution (EVOL) and international workshop on principles of software evolution (IWPSE)*, 2010, pp. 48–57.

[18] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *EMF: eclipse modeling framework*. Pearson Education, 2008.

[19] J. B. Warmer and A. G. Kleppe, *The object constraint language: getting your models ready for MDA*. Addison-Wesley Professional, 2003.

[20] C. J. Date and H. Darwen, "A Guide to the SQL Standard, vol. 3," 1987.

[21] "DIN SPEC 91345:2016-04 - Reference Architecture Model Industrie 4.0 (RAMI4.0)," Beuth Verlag, DIN, 2016.

[22] S. Feldmann, C. Legat, and B. Vogel-Heuser, "Engineering support in the machine manufacturing domain through interdisciplinary product lines: An applicability analysis," *IFAC-PapersOnLine*, vol. 28, no. 3, pp. 211–218, 2015.

## 5.2 Advanced CPPS Engineering Applications

### 5.2.1 Patterns for Reuse

[129] **K. Meixner**, A. Lüder, J. Herzog, D. Winkler, and S. Biffl. Patterns for reuse in production systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 31(11-12):1623–1659, 2021. doi: 10.1142/S0218194021400155

#### Aim

The publication "Patterns for Reuse in Production Systems Engineering" [129] introduces a modeling approach for PPR and capabilities as assets for reuse in domain engineering in a knowledge graph. The publication provides (i) requirements for a PPR asset graph representation with a dependency network, (ii) the meta-model of a knowledge graph for PPR assets, i.e, the Industry 4.0 Asset Network (iii) , for reuse in Production Systems Engineering (PSE) and (iv) four patterns for engineers to identify reusable assets for domain engineering.

#### Contribution to the thesis

This publication contributes to the research goals CPPS knowledge model (**G1**.), CPPS knowledge model reference architecture (**G2**.), CPPS knowledge model patterns (**G2**.), and CPPS reuse framework (**G2**.), .

This publication contributes to **RQ1**., **RQ2**. by addressing the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *enhanced quality and consistency* C4., and *facilitation of interdisciplinary collaboration* C5..

#### Abstract

In PSE, domain experts aim at reusing production processes implemented as Industry 4.0 assets and software. However, the knowledge on reusable assets is often scattered on multi-disciplinary engineering artifacts and domain experts, making it hard to find suitable reusable assets and map them to requirements. In this paper, we (i) identify challenges and requirements for reuse in PSE based on a domain analysis; (ii) introduce the I4AN that integrates multi-disciplinary dependencies between the assets and exposes recurring patterns; and (iii) present four patterns for reuse in PSE that aim at improving reuse efficiency and risk. We evaluate the I4AN with reuse scenarios in a feasibility study. The study results indicate that the I4AN model satisfies the elicited requirements and enables PSE domain experts to identify patterns for reuse in their contexts.

# Patterns for Reuse in Production Systems Engineering

Kristof Meixner*†, Arndt Lüder‡‖, Jan Herzog§, Dietmar Winkler*†, Stefan Biffl†‖
*Christian Doppler Laboratory SQI, †Inst. of Information Sys. Eng., TU Wien and ‖CDP, Austria
E-Mail: [first].[last]@tuwien.ac.at
‡Inst. of Ergonomics, Manufacturing Sys. and Automation, Otto-von-Guericke U., Germany
E-Mail: arndt.lueder@ovgu.de
§Department PPG-M/D, Volkswagen AG, Wolfsburg, Germany
E-Mail: [first].[last]@volkswagen.de

***Abstract*—In *Production Systems Engineering (PSE)*, domain experts aim at reusing production processes implemented as Industry 4.0 assets and software. However, the knowledge on reusable assets is often scattered on multi-disciplinary engineering artifacts and domain experts, making it hard to find suitable reusable assets and map them to requirements. In this paper, we (i) identify challenges and requirements for reuse in PSE based on a domain analysis; (ii) introduce the *Industry 4.0 Asset Network (I4AN)* that integrates multi-disciplinary dependencies between the assets and exposes recurring patterns; and (iii) present four patterns for reuse in PSE that aim at improving reuse efficiency and risk. We evaluate the I4AN with reuse scenarios in a feasibility study. The study results indicate that the I4AN model satisfies the elicited requirements and enables PSE domain experts to identify patterns for reuse in their contexts.**

***Keywords*—Reuse, Production Systems Engineering, Industry 4.0 asset, Industry 4.0 component.**

## I. Introduction

The *Industry 4.0 (I4.0)* initiative[1] has led to an increased focus on research related to *Production Systems Engineering (PSE)* in various research fields [1]. The I4.0 initiative envisions flexible and highly customizable production systems that interconnect modern manufacturing with the latest information and communication technology, so-called *Cyber-Physical Production Systems (CPPSs)* [2] that can self-adapt to particular conditions. These CPPSs incorporate *I4.0 assets* representing objects of perceived or actual value, such as products, processes, or resources [3]. The *Asset Administration Shell (AAS)*, their standardized digital representation [3], can describe their *skills* [4] and adapt the I4.0 assets to changes in the production environment. The aim is to fulfill business demands for increased flexibility and distribution of production, i.e., *production as a service*, and to react to shorter product life-cycles with reduced PSE project duration and effort [5].

These demands require the (partial) reuse of process and resource solutions from previous projects or standardized catalogues [6], [7]. Examples in automotive manufacturing are *position and screw* tasks, like screwing a dashboard into a car. In such cases, product type variants and their parameters vary, e.g., where and how tightly to screw which kind of screw to a dashboard. Yet, the processes and production resources executing these tasks, like robot arms, are quite similar. In addition, parts of the software controlling the resources and orchestrating the overall production system can be reused.

Reuse in PSE depends on efficiently identifying recurring patterns that can be integrated into a production system. These patterns need to follow reference architectures [8], [9] of (i) product types, e.g., car types, (ii) production processes, e.g., screwing processes, and (iii) production resource types and instances, e.g., screwing robots. Reuse also requires a pattern description on type and instance levels to facilitate referring to vendor catalogues or previous projects [6], [10].

The engineering of a production system is a collaborative effort of experts coming from many disciplines, like mechanical, electrical, and software engineering [11]. However, traditionally much of the engineering information is hidden in *scattered engineering artifacts* and much of the knowledge is *implicit domain knowledge* of engineering experts [5] (cf. Section IV). Furthermore, there is insufficient interdisciplinary exchange between the domains, leading to *hard to extract/collect/validate dependencies* from heterogeneous engineering artifacts and domain experts [12]. Hence, it is crucial in this multi-disciplinary environment to thoroughly model the (interdisciplinary) dependencies and boundaries in pattern analysis to reduce the risk of broken reusable assets.

Hence, we raise the main research question: *What approach can PSE experts use to efficiently identify patterns from existing engineering knowledge for reusing Industry 4.0 assets and related artifacts?*

In this paper, we (i) identify challenges and requirements for knowledge reuse in PSE, (ii) introduce the *Industry 4.0 Asset Network (I4AN)*, a model to integrate the scattered knowledge and enable engineers to identify patterns for reuse to improve the effectiveness and efficiency of the PSE life-cycle, and (iii) present four recurring high-level patterns in PSE as a basis for identifying applied solution patterns for similar problems. We evaluate these contributions with an instance of an I4AN. Therefore, we investigate to what extent typical reuse scenario questions can be answered as queries to the I4AN.

The remainder of this paper is structured as follows: Sec-

DOI: 10.18293/SEKE2021-150
[1]Industry 4.0 Initiative: https://www.plattform-i40.de

tion II summarizes related work. Section III presents the research questions and method. Section IV introduces an illustrative use case and identifies requirements for reuse in PSE. Section V introduces the *Industry 4.0 Asset Network (I4AN)* for reuse in PSE and four high-level reusable patterns. Section VI reports on a feasibility study to evaluate the I4AN capabilities and discusses the results and limitations of the research. Section VII concludes and outlines future work.

## II. Related Work

This section summarizes related work on *Production Systems Engineering (PSE)*, knowledge management, and reuse.

### A. Production Systems Engineering

PSE is a multi-disciplinary process that involves various disciplines, like mechanical, electrical, and software engineering [11]. Engineering teams iteratively perform tasks, like mechanical design or implementation of the control software [13], to engineer the desired production system.

In PSE, engineers create various types of engineering artifacts and models [5], [14]. However, the used formats and tools have traditionally been optimized for a single discipline, and while engineers are well connected within their domains, there is often an insufficient interdisciplinary exchange. Further, the engineering artifacts and information are scattered throughout the engineering landscape [5]. Much of the engineering knowledge is implicit knowledge of the domain experts. These issues pose an increasing challenge related to information management and reuse within PSE projects [15].

Yet, for the suitable and correct production system design, it is crucial to exchange information and knowledge between the disciplines effectively and efficiently [16]. In addition, the reuse of designs and specifications for recurring problems, e.g., using a robot type for similar tasks, improves the quality and helps to reduce PSE project duration, effort, and risk [8].

In this paper, we introduce a network, based on explicitly linked assets and artifacts, to provide the foundation for domain experts to link their specific knowledge representations.

Industry 4.0 (I4.0) addresses the overall digitalization and networking of production system elements, i.e., I4.0 assets, towards *Cyber-Physical Production Systems (CPPS)*. I4.0 assets are physical or immaterial objects of perceived or actual value [3]. An increasing focus can be recognized on product, process, and resource-related I4.0 assets, which we mainly refer to. An *Asset Administration Shell (AAS)* provides a digital representation [3] of I4.0 assets with their property views and skills. These descriptions should include the information and knowledge for an automated orchestration, which requires explicit knowledge on the I4.0 assets and their dependencies.

Pfrommer *et al.* [17] define a skill as the ability of a resource to perform a process, while a production skill gives the requirements [4]. Candidio *et al.* [18] understand a skill as ability to perform actions that are needed to support the production process. Meixner *et al.* [10] described how to abstract skills of resources from process requirements. Hence, models must represent the *required skills* of processes and the *provided skills* of resources [17], [19]. However, the identification of reusable I4.0 Asset candidates requires representing skills as I4.0 Assets to provide the abstraction for the digital representation of boundaries between reusable patterns.

In this paper, we represent *skills* for the first time as I4.0 Asset, as an abstraction between processes and resources to foster the identification of reusable I4.0 Asset candidates.

### B. Modeling Engineering Knowledge in PSE

Sabou *et al.* [20] introduced a knowledge graph for reuse in the software engineering domain, but without multi-model links that facilitate reuse in PSE.

For modeling Cyber-Physical Production Systems (CPPSs), two main research and development directions have been pursued. First, IT systems engineering uses CASE tools based on UML [21]. As a result, systems engineering methodologies have been created utilizing domain-crossing modeling standards like SysML[2] [22]. Second, engineering data exchange preserves the multi-model nature of PSE knowledge and builds on standardized data formats like *AutomationML*[3] [16] to make data integration more efficient. In this paper, we build on cross-linking assets and engineering artifacts as a basis for an improved reuse considering dependencies.

Both directions require for explicitly modeling PSE knowledge to reflect the specifics of this domain, including (i) modeling *part-whole* relations, (ii) *connections between components* [23], and (iii) *technical dependencies* of the various involved technical disciplines [24] The authors observed in PSE containment hierarchies to be well-established and frequently used to organize assets in PSE models. Furthermore, discipline-specific dependencies are often represented in discipline-specific models as interfaces.

Feldmann *et al.* [25] introduced an approach for managing inconsistencies in a multi-disciplinary multi-model environment using links between objects in PSE. However, the approach by Feldmann *et al.* [25] does not consider I4.0 assets and skills as first-class citizens in PSE. However, this integration is a foundation for better identifying reusable assets based on the digital representation of their skills. In this paper, we build on their meta-model [25] to integrate links between I4.0 assets coming from several engineering disciplines.

### C. Reuse in Production Systems Engineering

Main approaches to reuse are (i) *clone and own* [26] and (ii) *reuse of components*, such as software libraries. However, these general reuse approaches do not sufficiently cover requirements in multi-disciplinary environments, like PSE.

In PSE, several reference frameworks address the reuse of assets. The guideline VDI 2206 [27] describes the V-Model as a procedure for structured PSE. It encourages to reuse requirements and partial implementations in later phases, like the test phase, without mentioning how. Jazdi *et al.* [8] provided first methodologies related to the systematic identification of

[2]SysML: https://www.sysml.org
[3]AutomationML: https://www.automationml.org

reusable system components. Most of them are based on the idea of mechatronic systems [28] following the VDI 2206 [27].

The guideline VDI 3695 [29] understands reuse as a method for engineering optimization and defines five types of reuse, i.e., *Reuse Levels (RL)*: Reuse (i) by employees on their own accord (RL-A), (ii) controlled within the project (RL-B), (iii) controlled from a central point across all projects (RL-C), (iv) based on a reference model (RL-D), and (v) based on internal and external standards (RL-E). The effectiveness and efficiency of reuse of assets depends on the level of reuse maturity [29] and the relations between assets [9]. Yet, this information is insufficiently available in PSE due to scattered artifacts and information.

In software engineering, one specific domain in the PSE engineering process, *design patterns* [30] are a widely adopted standard for reuse. Design patterns aim at developing software faster and in better quality while reducing risks and cost [30]. Therefore, design patterns provide adaptable design solution templates to general problems that software developers face. A design pattern consists of (i) a pattern name, (ii) a description of a problem that should be solved, (iii) a solution description with its elements and dependencies, and (iv) implications of the pattern, such as benefits and limitations. Software design patterns can serve as a blueprint for PSE design patterns but need to be adapted to the multi-disciplinary context of PSE.

In this paper, we build on the guideline VDI 3695 to describe the reuse maturity in PSE and the idea of mechatronic units as reusable entities [28]. Furthermore, we build on *design patterns* as a concept to identify reusable patterns in PSE.

## III. Research Questions and Approach

In this paper, we follow the *Design Science* methodology [31] to investigate how to improve identifying I4.0 assets for reuse in PSE. Therefore, we (i) conducted a domain analysis in the automotive industry, (ii) condensed a representative use case, and (iii) elicited requirements on I4.0 asset reuse with domain experts at medium-to-large European PSE companies (cf. Section IV).

Considering identified gaps in the related work and requirements in PSE, we formulate the following research questions.

**RQ1a.** *What model and elements facilitate identifying I4.0 assets for reuse in PSE?* The systematic reuse of I4.0 assets in engineering fosters quality and efficiency [8]. However, in PSE, the knowledge required for reuse often consists of heterogeneous information and implicit knowledge, scattered across the engineering landscape. To address RQ1a, we investigated recurring engineering artifacts from the domain analysis to identify knowledge elements that help engineers in efficiently identifying reusable assets. Our contribution is the *Industry 4.0 Asset Network (I4AN)* as a foundation to explore assets suitable for reuse.

**RQ1b.** *What connections between system parts and engineering artifacts represent dependencies in an I4.0 asset network as a foundation for identifying sets of reusable assets?* Connections and relationships between I4.0 assets provide data to understand internal and external dependencies of CPPS assets. These dependencies are crucial to coherently identify and explain which potentially reusable assets can be reused *as-is* or require further assets to be included to correctly reuse them. To address RQ1b, we build on the *Industry 4.0 Asset Network (I4AN)* , coming from *RQ1a*, and investigated which links represent internal and external dependencies that are relevant to facilitate the reuse of assets. Our contributions focus on the classification of dependencies in the I4AN that are crucial to identify sets of reusable assets.

**RQ2.** *Which basic patterns for reuse facilitate identifying best-practice pattern candidates for PSE?* For identifying patterns for reuse, engineers require a starting point in their particular context. For instance, engineers are likely to recognize a pattern as an initial set of assets and their dependencies to other assets. Basic patterns, which occur independently from the particular context of the PSE project, can represent such a starting point. From domain analysis and discussions with engineers, we identify basic patterns that regularly occur in PSE. These patterns provide blueprints to help engineers identify reusable assets in I4AN instances.

Each research question addresses parts of the overarching question (cf. Section I) tying together model elements, their dependencies, and patterns for the efficient reuse of CPPS knowledge. We evaluate the I4AN in a feasibility study for the use case *"Car Body with Screwed-on Parts"* (cf. Section IV). Therefore, we use data from a sample of artifacts from the domain analysis. We investigate to what extent advanced reuse scenario questions can be answered by queries to the I4AN.

## IV. Illustrative Use Case

This section introduces the use case *"Car Body with Screwed-on Parts"*. We condensed the use case from a domain analysis in the automotive manufacturing domain. The analysis was conducted in a setting with 80 types of screwing robot cells and 27 robot types.

In automotive manufacturing, human workers collaborate with industrial robots in mounting lines to place and screw various components onto a car body using screwdrivers. Typical mid-class cars contain screws of 80 screw types at 1,500 to 1,800 screw positions. Figure 1 shows the use case with its I4.0 assets and their connections. The left-hand side shows a screwing process consisting of two steps: (i) *positioning* the dashboard and the screws and (ii) *fastening* the screws. Both steps are characterized by process requirements, defining the necessary skills of the resources including technical or economic parameters. In PSE, relevant resources, i.e., resource hierarchies, (see right-hand side of Figure 1) are selected and orchestrated to provide the required skills [17], [19].

In theory, one can engineer an optimized robot-screwdriver combination for each screw type to maximize production effectiveness and efficiency. Yet, this approach might lead to around 80 different robot and screwdriver types, adding significant costs for installation, maintenance, and expert knowledge.

In practice, PSE aims at cost-optimized system designs [32]. Hence, a sufficiently effective and efficient robot-screwdriver combination to each screw type can be assigned, minimizing

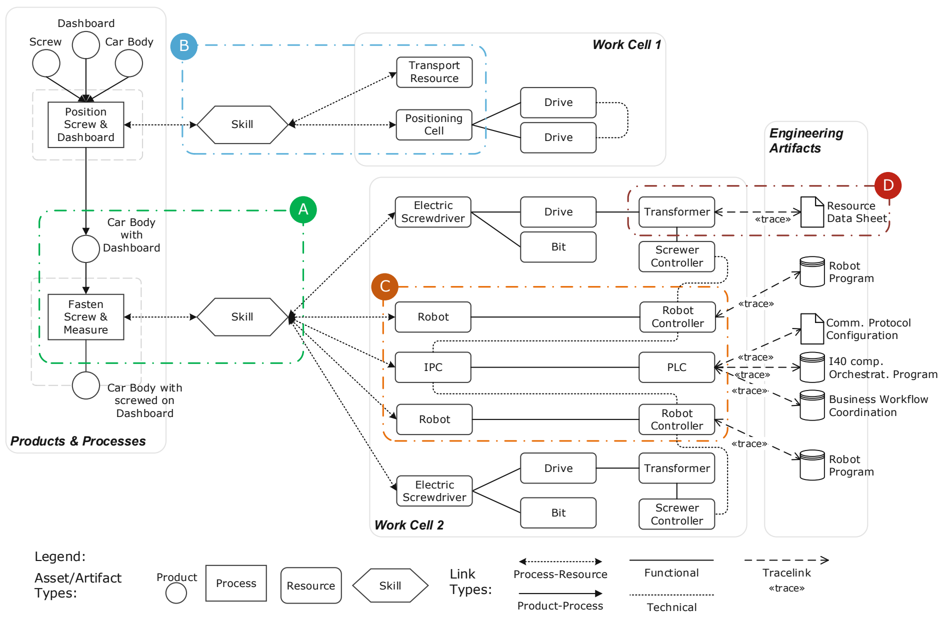


Fig. 1. *Reuse Patterns* (dashed boxes) in an asset network for the use case *"Car Body with Screwed-on Parts"* (in adapted VDI 3682 notation [10], [29]).

the number of robot types and investments in spare parts and know-how. This approach may significantly reduce costs in comparison to a high-variety approach.

Identifying an optimized set of robot-screwdriver combinations for the high number of different screws types and positions requires the identification of (i) *resources* that can execute several varying screwing tasks and (ii) *engineering artifacts* that can be reused, such as control programs. However, identifying suitable solutions is difficult due to the scattered information and the engineers' implicit knowledge.

Achieving these advantages requires a set of reusable patterns. This set of patterns can be completed by (i) *identifying similar components* within existing engineering projects, (ii) *mapping* these components to expected future requirements, and (iii) *abstracting* these components with respect to possible adaptations for application-case related parameterization [8].

**Requirements for I40 Asset Reuse.** From the domain analysis, we elicited the following *requirements Rx* towards asset reuse in PSE with eight domain experts from five medium-to-large European PSE companies[4].

**R1. I40 Asset Map.** Domain experts require an *Industry 4.0 Asset Map*, i.e., an overview on the assets in the planning phase to explicitly represent implicit knowledge and relevant information as a context for reuse, currently scattered across various engineering artifacts. This requirement is adapted from software engineering, i.e., documenting the project structure and software artifacts, to multi-disciplinary PSE assets in the *Product-Process-Resource (PPR)* scope.

**R2. I40 Dependency Network.** As a basis to identify patterns for reuse in PSE, domain experts require an explicit representation of the links and dependencies of and between assets coming from several engineering disciplines. This concerns mainly three different views. Product engineering requires links between product components, processes, and their required skills that a process requires from a resource to automate the process. Systems engineering concerns relationships between resources and their provided capabilities. The assets and dependencies can be represented in an *Industry 4.0 Dependency Network* that adds information and knowledge required for reuse to the *Industry 4.0 Asset Map*.

**R3. System Boundary.** For reuse, a system or subsystem containing the reusable assets needs to have a clearly defined boundary. System boundaries are a means to group assets into a meaningful set of assets that can be reused. A boundary also allows to investigate incoming and outgoing dependencies.

[4]The experts rated their company's maturity level of asset reuse, using the VDI 3695 classification, at reuse levels RL-C (controlled from a central point) or RL-E (reuse based on internal and external standards).

Thus, system boundaries serve as a basis for systematically reusing (parts of) a solution that was used in previous projects. Without a clear boundary, it is unclear which elements can, should, or have to be included in a set of reusable assets. Furthermore, system boundaries enable developing and using metrics, like complexity, to compare patterns.

**R4. Solution Design Abstraction.** As a foundation to identify reusable patterns, domain experts need a representation of solution design candidates at a suitable level of abstraction. This abstraction is required to allow the adaptability and portability of a pattern to similar problems with varying characteristics. For example, to make a solution for a *position* task reusable requires hiding unnecessary attributes and dependencies. In the use case, the robot positioning accuracy is a relevant characteristic, while the way how the robot moves might be irrelevant. *Solution Design Abstraction* facilitates (i) generalizing from a particular solution instance to a more general level of problems and (ii) finding reusable solution candidates in similar or historic designs.

The following section builds on this use case to illustrate a novel knowledge representation model for Industry 4.0 Assets for identifying patterns for reuse.

## V. Patterns for Reuse in PSE

This section presents the Industry 4.0 Asset Network and four basic patterns to identify concrete patterns for reuse.

### A. I40 Asset based Network with Dependencies

To address *RQ1a* and *RQ1b*, we investigated the data of robot cells with up to two robots from the use case context with domain experts. From this data, we determined knowledge elements that we can use for identifying abstract patterns for reuse. These elements were used to build a condensed metamodel as the foundation for the I4AN. This section illustrates the metamodel and the I4AN using the *car body with screwed-on parts* use case from Section IV.

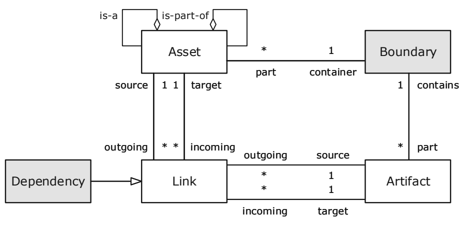


Fig. 2. *Asset*, (Engineering) *Artifact* and *Link* meta model, based on [33]

Figure 2 shows the metamodel (in UML notation) containing the *Asset* class, one of the Industry 4.0 Asset types *product*, *process*, *resource*, or *skill*. An *Asset* can be a specialization (*is-a* relation, e.g., an electric screwdriver is a type of screwdriver) and/or a part (*is-part-of* relation, e.g., a bit is part of a screwdriver) of another asset. An *Artifact* is an engineering object created during design time, e.g., an electrical plan or robot program, or during runtime, e.g., a set of qualitative data. A particular *Link* can connect assets with each other or to artifacts. *Links* can have different forms (cf. Figure 1) realized using typed properties (not shown in the meta-model): *Functional links* between production resources may represent a resource composition. *Technical links* may represent a wired connection from an *Industrial PC (IPC)* to a robot. To model a connection between an *Asset* and an *Artifact*, we use *Trace Links*, e.g., a robot controller requires a robot program. A *Link* can be manifested as *Dependency*, if the link is strictly required by an *Asset*. *Assets*, *Artifacts*, and *Links* can have attributes that describe characteristics of the particular object. These properties follow the I4.0 Asset Administration Shell (AAS) [3] design to facilitate the standardized representation of property views coming from several engineering disciplines. A *Boundary* object represents a pattern boundary that contains *Assets* and *Artifacts*, e.g., boundary *(A)*.

These concepts provide the foundation to build an I4AN that explicitly represents PSE information and knowledge for a wide range of applications, such as change impact analysis. Figure 1 illustrates an I4AN with the relevant engineering artifacts and the links between the assets. This model can be created automatically by exploiting appropriate engineering data logistic systems [12]. This overall model can be the starting point to identify common reusable patterns [19].

### B. Patterns for I40 Asset Reuse

This section describes four basic patterns for identifying best-practice candidates for reuse in their context. These identification patterns can be used as a starting point to identify patterns in the particular PSE contexts of domain experts.

The reuse of assets requires considering the asset itself and, beyond that, its embedding in the surrounding system and functional intentions [6], [9], [28]. As described, PSE comprises two main phases, rough and detail planning.

The rough planning phase consists of matching process skills required by products and provided by resources. This comparison shall be based on *product creation* (P1) and *process execution* (P2) patterns.

**P1. Product-Process-Skill Pattern.** Product creation in PSE aims at providing the combination of products with their requirements and processes to manufacture them. ***Aim:*** The *Product-Process-Skill* pattern (cf. Figure 1, tag *A*) supports product engineers in selecting appropriate processes for their products. This product creation pattern contains production processes with their input and output materials, boundary conditions, and required skills. ***Solution:*** The pattern can be identified by collecting all assets connected to the related processes by product-process-related links: For an output product isolate the input products and determine their relevant properties. For each input product determine the required process steps and build the aggregated required skills of the steps according to [10]. Group the products, process steps, and skills into a boundary object. For the outgoing and incoming links, determine whether they are strict dependencies. For dependencies, decide if you need to either expand the boundary or create a depending pattern object. ***Example:*** An example are screw-screwing combinations. We identified different reuse patterns from equivalence classes based on the screwing bit, the applicable torque, and the screw material (magnetic vs. non-magnetic) with industry partners.

**P2. Skill-Resource Pattern.** Process execution in PSE is to identify resources able to execute a production process based on their functional skills. ***Aim:*** The *Skill-Resource* pattern (cf. Figure 1, tag *B*) helps to select appropriate resources matching to the *Product-Process-Skill* pattern. This process execution pattern contains resources with their properties, boundary conditions, and provided skills. ***Solution:*** The pattern can be identified by collecting all assets connected to the related resource links: For a set of connected resources, determine their provided skills and properties. From the skills, build the aggregated provided resources skills according to [10]. Group the resources and skills into a boundary object and determine the dependencies. For dependencies, decide if you need to either expand the boundary or create a depending pattern. ***Example:*** The pattern supports the definition of skills, e.g., positioning, with predefined attributes, like positioning accuracy, which are fulfilled by a set of resources.

The main concern within the detail planning phase is realizing production resources providing all necessary functionalities to fulfill the required skills. Here, patterns related to resource structuring and functionality are relevant. Thus links shall be considered depending on the use case.

**P3. Resource-Resource Composition Pattern.** The goal of detailed engineering in PSE is detailing and programming the selected resources. ***Aim:*** The *Resource-Resource composition* pattern (cf. Figure 1, tag *C*) represents the composition of a resource from sub-components, with the knowledge on technical parameters and dependencies on the type and instance levels. A quality ensured resource tree pattern could be applied at this point, reflecting the optimized orchestration of resources. ***Solution:*** For a group of connected resources (*part-of* relation) determine which resources are required to either fulfill a particular skill or if they require each other for functionality. Group the strictly required resources into a boundary. For dependencies, decide if you need to either expand the boundary or create a depending pattern object. ***Example:*** Screwdrivers can be driven, e.g., electrically or pneumatically. Depending on the drive, the screwdriver requires a transformer for the current or not, which can be expressed in an *RR* pattern.

**P4. Resource-Artifact Pattern.** Within the commissioning phase of PSE the detailed resource system is established according to the relevant engineering artifacts, e.g., relevant for operation. ***Aim:*** The *Resource-Artifact* pattern (cf. Figure 1, tag *D*) aims at binding the required engineering artifacts to the resources used in the production system. This helps engineers to reuse resources and their corresponding data or programs as a bundle. ***Solution:*** From a resource, follow the trace links to the engineering artifacts. For the resource and the necessary engineering artifacts, use a boundary object to group them. For incoming or outgoing dependencies from resources or engineering artifacts, decide whether to expand the boundary or create a depending pattern object. ***Example:*** Screwdrivers have a minimum, maximum, and yield torque for a screwing process. The screwdrivers and function blocks controlling the torque of the screwdrivers can be expressed as a pattern and reused in future projects.

The use case *Car Body with screwed-on parts* can benefit from reuse patterns in (at least) four ways: (i) The *product-process-skill* pattern can support product engineers in selecting appropriate screwing processes for their car body parts (see tag *A* in Figure 1). (ii) The *skill-resource pattern* facilitates selecting appropriate screwdrivers to screwing processes (see tag *B* in Figure 1). (iii) The *resource-resource composition* pattern can be applied for the optimized combination of screwing resources, e.g., robots and robot controllers (see tag *C* in Figure 1). (iv) The *resource-artifact* pattern can be applied for reusing engineering artifacts, e.g., robot controllers and robot control programs (see tag *D* in Fig. 1).

## VI. Feasibility Study and Discussion

This section presents a preliminary feasibility study and discusses the contributions with a focus on the research questions raised in Section III.

### A. Preliminary Feasibility Study

As a proof of concept, we used a part of the production system for the investigated use case *"Car Body with Screwed-on Parts"* from the initial domain analysis to design and instantiate the *Industry 4.0 Asset Network (I4AN)* in a *Neo4J*[5] graph database. The I4AN was found easy to extract from existing engineering information, which has to be integrated according to the the I4.0 AAS design [3].

The graph database facilitated the effective and efficient exploration, querying, and visualization of the linked assets. In addition to the *technical links* between assets coming from engineering models, we instantiated *dependency links* between the assets. Deep domain expert knowledge has to be added to the I4AN manually. The concepts in the I4AN facilitated adding previously implicit domain knowledge to the graph.

The I4AN instance associated to Figure 1 enables identifying I4.0 Assets that belong to a pattern for I4.0 Asset reuse (cf. Section VI-B). To investigate the functionality, we issued queries onto the I4AN to track the dependencies. We used iterative queries, similar to cause-effect graph exploration [33], starting at a selected I4.0 Asset, such as a skill, and followed the multi-model links to neighboring assets of a specified type until reaching a stopping condition. We were able to efficiently isolate parts of the I4AN that correspond to the basic patterns introduced in Section VI-B. This approach also worked for the reuse scenario *system boundary analysis* that can be translated into the question: *Which set of dependency links connects a selected set of assets to their immediate neighboring assets?* This capability indicates that engineers can utilize the I4AN to investigate the network to identify familiar patterns of assets as candidates for reuse.

[5]Graph database *Neo4J*: https://neo4j.com

### B. Discussion

We conducted a domain analysis with 80 types of robot cells and 27 robot types. Further, we elicited requirements from domain experts at five European PSE companies. The requirements showed that a key aspect is modeling the multi-disciplinary dependencies between assets and engineering artifacts that need to be considered to identify reusable asset patterns. It is also essential to thoroughly model the boundaries of the patterns to allow suitable reuse in practice among the involved engineering disciplines.

**RQ1a** and **RQ1b** concerned models and dependencies that facilitate the identification of assets suitable for reuse. To address RQ1a and RQ1b, Section V-A introduced the *Industry 4.0 Asset Network (I4AN)* that addresses requirements *R1* to *R3* identified in Section IV. The I4AN builds on I4.0 assets and uses their administration shell to integrate property views from several engineering disciplines. In comparison to patterns in software engineering, this multi-disciplinary aspect adds complexity to identifying patterns for reuse in PSE.

We go beyond the state of the art [4], [18] by modeling *skills* as I4.0 assets using their digital representation for linking multi-disciplinary assets and identifying boundaries for reusable assets. We build on and go beyond [25] by integrating multi-disciplinary multi-model links between I4.0 Assets.

**RQ2** asked which basic patterns for reuse facilitate the identification of patterns for reuse. To address RQ2, Section identified four basic patterns addressing requirement *R4* (cf. Section IV). These patterns specifically incorporate regularly occurring connected assets in PSE that can be reused for similar problems. Therefore, they provide guidance for reuse design and management with the I4AN. In this sense, the I4AN provides designers with the capability to describe partial solutions and integrate partial solutions into a complete solution from production processes to automation devices that automate the production process.

**Limitations.** The following limitations require further investigation. The research in this paper focused on the reuse of production processes and associated automation system elements in a typical use case of automotive manufacturing, the *Car Body with Screwed-on Parts* use case. As we assume the findings of this paper to be relevant in the broader scope of production processes and automation system elements, e.g., for discrete production and continuous production, the approach should be investigated in a broader range of application areas.

The *domain analysis* was conducted by one of the paper authors with consultation from domain experts and checked for plausibility by the author team. While the *feasibility study* focused on a I4AN for a robot cell of typical complexity, the authors of this paper, consulting with domain experts in car manufacturing, conducted the design of the I4AN including dependencies that are missing in traditional PSE design. This reflects the current practice of PSE engineering only partially and introduced bias to the study, requiring validation in a range of traditional and advanced PSE environments.

## VII. Conclusion and Future Work

The Industry 4.0 (I4.0) vision of production systems that are easy to adapt depends on advanced capabilities for reusing proven production processes, I4.0 assets and software-intensive components that automate these production processes. In Production Systems Engineering (PSE), the reuse of I4.0 assets requires understanding the dependencies of these assets in multi-disciplinary systems-of-systems engineering with heterogeneous models.

This paper investigated the information requirements for advanced multi-disciplinary reuse scenarios, such as *process and resource identification* and for *system boundary analysis*. To address the challenges of scattered and implicit domain expert knowledge that may lead to overlooking risky dependencies of reusable system elements, we introduced the *Industry 4.0 Asset Network (I4AN)*. The I4AN builds on the I4.0 Asset Administration Shell [3] design to integrate system element properties and dependencies from several engineering disciplines, such as mechanical, electrical, and software interfaces and technical links.

Therefore, the I4AN enables designing a knowledge graph that represents for a reuse scenario important multi-disciplinary dependencies between system elements as neighborhoods of I4.0 Assets. Further, the I4AN concepts facilitate representing domain expert knowledge that was implicit, e.g., to recommend using a resource type with a process type.

We presented the use case *"Car Body with Screwed-on Parts"* to illustrate typical I4.0 Assets and links in production processes and robot cells widely used in car manufacturing. In the I4AN of the use case (cf. Figure 1), we identified four types of patterns for reuse.

In a feasibility study, we evaluated the I4AN with reuse scenarios by instantiating an I4AN knowledge graph formulating scenario concepts and questions as data in and queries to the knowledge graph. The study results indicate that the I4AN model is a good foundation for PSE domain experts to identify patterns for reuse in their contexts.

The research results advance the state of the art in knowledge engineering in PSE by modeling the *Skill* concept as an I40 Asset. The I4AN provides a lens for analyzing similarities and differences in production process and system designs. To this end, we are providing the foundations for advanced reuse design and management with the I4AN and patterns.

The research results advance the state of the art by adapting blueprints for design pattern to a multi-disciplinary engineering environment where multi-model links are crucial. The I4AN provides designers with the capability to describe partial solutions and integrate these partial solutions into a complete solution, from production process to automation devices that automate the production process. The I4AN facilitates identifying risky external systems dependencies across several engineering disciplines as input to assess the reuse effort and risk of candidate solution designs.

**Future Work.** *Validation of patterns for reuse.* We plan to investigate I4AN applications for reuse to improve PSE tools, e.g., with knowledge on multi-model dependencies.

*Scalability.* We see the need to investigate the scalability of the I4AN in a larger context and with additional engineering disciplines to evaluate the impact on the multi-disciplinary dependencies and boundaries beyond the scale of work cells.

*Skills.* We consider examining the extended use of *skills* as an advanced method to abstract from process requirements to resource capabilities and their role in reusable process and resource assets, e.g., using standardized catalog search.

*Extension of the I4AN with Semantic Web content.* For the PSE domain, the I4AN seems well represented in a graph database as this technology is increasingly well accepted in PSE, while Semantic Web technology is mainly used in research. We envision extending the I4AN with knowledge organized with Semantic Web technologies, e.g., issues, recommendations as natural text. The I4AN knowledge graph can collect knowledge instances that can be converted efficiently to Semantic Web technologies to facilitate research on industrial data for Semantic Web researchers.

*Security.* Aggregating domain knowledge in an I4AN creates a high-value knowledge graph. This graph requires research on security concerns, e.g., theft of intellectual property or using it to plan attacks on systems that represent critical infrastructure.

## Acknowledgment

The financial support by Christian Doppler Research Association, Austrian Federal Ministry for Digital and Economic Affairs, and National Foundation for Research, Technology and Development is gratefully acknowledged. The competence center CDP is funded within the framework of COMET – Competence Centers for Excellent Technologies by BMVIT, BMDW, and the federal state of Vienna, managed by the FFG.

## References

[1] B. Vogel-Heuser, T. Bauernhansl, and M. Ten Hompel, “Handbuch Industrie 4.0 Bd. 4,” *Allgemeine Grundlagen*, vol. 2, 2020.

[2] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda, “Cyber-physical systems in manufacturing,” *CIRP Annals*, vol. 65, no. 2, pp. 621 – 641, 2016.

[3] Plattform Industrie 4.0 and ZVEI, “ Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01 Review),” German BMWI, Standard, Nov. 2020, https://bit.ly/37A002I.

[4] J. Pfrommer, D. Stogl, K. Aleksandrov, V. Schubert, and B. Hein, “Modelling and orchestration of service-based manufacturing systems via skills,” in *19th IEEE ETFA*, 2014, pp. 1–4.

[5] A. Lüder, N. Schmidt, K. Hell, H. Röpke, and J. Zawisza, *Identification of Artifacts in Life Cycle Phases of CPPS*. Cham: Springer International Publishing, 2017, pp. 139–167.

[6] H. Johannesson, *Emphasizing Reuse of Generic Assets Through Integrated Product and Production System Development Platforms*. New York, NY: Springer New York, 2014, pp. 119–146.

[7] A. Lüder, N. Schmidt, K. Hell, H. Röpke, and J. Zawisza, *Fundamentals of Artifact Reuse in CPPS*. Cham: Springer International Publishing, 2017, pp. 113–138.

[8] N. Jazdi, C. R. Maga, P. Göhner, T. Ehben, T. Tetzner, and U. Löwen, “Improved Systematisation in Plant Engineering and Industrial Solutions Business - Increased Efficiency through Domain Engineering,” *Automatisierungstechnik*, vol. 58, no. 9, pp. 524–532, 2010.

[9] F. Stallinger, R. Neumann, R. Plosch, P. Hehenberger, B. Bohm, A. Kohlein, and N. Gewald, “Improving mechatronical engineering: An artifact-assessment-based approach,” in *16th IEEE ETFA*. IEEE, 2016, pp. 813–820.

[10] K. Meixner, A. Lüder, J. Herzog, H. Röpke, and S. Biffl, “Modeling expert knowledge for optimal CPPS resource selection for a product portfolio,” in *25th IEEE ETFA*. IEEE, 2020, pp. 1687–1694.

[11] A. Lüder, N. Schmidt, K. Hell, H. Röpke, and J. Zawisza, *Description means for information artifacts throughout the life cycle of CPPS*. Cham: Springer International Publishing, 2017, pp. 169–183.

[12] A. Lüder, A. Behnert, F. Rinker, and S. Biffl, “Generating industry 4.0 asset administration shells with data from engineering data logistics,” in *25th IEEE ETFA*. IEEE, 2020, pp. 867–874.

[13] A. Strahilov and H. Hämmerle, “Engineering Workflow and Software Tool Chains of Automated Production Systems,” in *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 2017.

[14] K. Paetzold, *Product and Systems Engineering/CA* Tool Chains*. Cham: Springer International Publishing, 2017, pp. 27–62.

[15] R. Drath, M. Barth, and A. Fay, “Offenheitsmetrik für engineering-werkzeuge,” *atp edition*, vol. 54, no. 09, pp. 46–55, 2012.

[16] A. Lüder, N. Schmidt, and R. Drath, *Standardized Information Exchange Within Production System Engineering*. Cham: Springer International Publishing, 2017, pp. 235–257.

[17] J. Pfrommer, M. Schleipen, and J. Beyerer, “PPRS: Production skills and their relation to product, process, and resource,” in *18th IEEE ETFA*. IEEE, 2013, pp. 1–4.

[18] G. Cândido and J. Barata, “A multiagent control system for shop floor assembly,” in *Holonic and Multi-Agent Systems for Manufacturing*, V. Mařík, V. Vyatkin, and A. W. Colombo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 293–302.

[19] J. Herzog, H. Röpke, and A. Lüder, “Allocation of PPRS for the plant planning in the final automotive assembly,” in *ETFA*. IEEE, 2020, pp. 813–820.

[20] M. Sabou, F. J. Ekaputra, T. Ionescu, J. Musil, D. Schall, K. Haller, A. Friedl, and S. Biffl, “Exploring enterprise knowledge graphs: A use case in software engineering,” in *European Semantic Web Conference*. Springer, 2018, pp. 560–575.

[21] G. Booch, *The Unified Modeling Language Use Guide*. Addison-Wesley Object Technology, 2017.

[22] D. F. Eugenio Brusa, Ambra Calà, Ed., *Systems Engineering and Its Application to Industrial Product Development*. Springer, 2018.

[23] C. Legat, C. Seitz, S. Lamparter, and S. Feldmann, “Semantics to the shop floor: Towards ontology modularization and reuse in the automation domain,” *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 3444–3449, 2014, 19th IFAC World Congress.

[24] U. Lindemann, M. Maurer, and T. Braun, Eds., *Structural complexity management: An approach for the field of product design*. Springer, 2009.

[25] S. Feldmann, M. Wimmer, K. Kernschmidt, and B. Vogel-Heuser, “A comprehensive approach for managing inter-model inconsistencies in automated production systems engineering,” in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, 2016, pp. 1120–1127.

[26] S. Fischer, L. Linsbauer, R. E. Lopez-Herrejon, and A. Egyed, “Enhancing Clone-and-Own with Systematic Reuse for Developing Software Variants,” in *2014 IEEE International Conference on Software Maintenance and Evolution*, sep 2014, pp. 391–400.

[27] *VDI Guideline 2206: Design methodology for mechatronic systems*, VDI-Verlag, Düsseldorf Std., 2004.

[28] A. Lüder, L. Hundt, M. Foehr, T. Holm, T. Wagner, and J. Zaddach, “Manufacturing system engineering with mechatronical units,” in *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, 2010, pp. 1–8.

[29] *VDI Guideline 3695: Engineering of industrial plants - Evaluation and optimization*, Beuth Verlag Std., 2009.

[30] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[31] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.

[32] R. Mehr and A. Lüder, *Managing Complexity Within the Engineering of Product and Production Systems*. Cham: Springer International Publishing, 2019, pp. 57–79.

[33] S. Biffl, A. Lüder, K. Meixner, F. Rinker, M. Eckhart, and D. Winkler, “Multi-view-model risk assessment in cyber-physical production systems engineering,” in *MODELSWARD*. SCITEPRESS, 2021, pp. 163–170.

### 5.2.2 Patterns for Reuse – Extension

#### Citation

[128] **K. Meixner**, A. Lueder, J. Herzog, D. Winkler, and S. Biffl. Patterns for reuse in production systems engineering. In S.-K. Chang, editor, *The 33rd International Conference on Software Engineering and Knowledge Engineering, SEKE 2021, KSIR Virtual Conference Center, USA, July 1 - July 10, 2021*, pages 1623–1659. KSI Research Inc., 2021. doi: 10.18293/SEKE2021-150

#### Aim

The publication "Patterns for Reuse in Production Systems Engineering" [128] extends the previous publication with an in-depth investigation of the asset network and a feasibility study. The publication provides (i) an extended knowledge graph meta-model for PPR assets with capabilities and skills, i.e, the Industry 4.0 Asset Network (I4AN), for reuse in PSE including a more detailed asset dependency network serving as a reference model for industry, (ii) an elaborate description of the patterns, (iii) a feasibility study with a realization of the approach in the Neo4J graph database with queries for stakeholder questions.

#### Contribution to the thesis

This publication contributes to the research goals CPPS knowledge model (**G1**.), CPPS knowledge model reference architecture (**G2**.), CPPS knowledge model patterns (**G2**.), CPPS reuse method (**G2**.), CPPS knowledge model prototype (**G4**.), and case study evaluation (**G5**.).

This publication contributes to **RQ1**., **RQ2**. by addressing the VDI 3695 measures of *models and description languages* M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *enhanced quality and consistency* C4., and *facilitation of interdisciplinary collaboration* C5..

#### Abstract

In PSE, domain experts aim at reusing partial system designs implemented as Industry 4.0 assets and software. However, the knowledge on assets is often scattered across engineering artifacts from multiple disciplines and domain experts, making it difficult to find reusable assets and map them to requirements. In this paper, we (i) identify challenges and requirements for the representation of reuse knowledge in PSE, based on the results of a domain analysis in automotive manufacturing; (ii) refine the I4AN meta-model that integrates multi-disciplinary dependencies between the assets; (iii) introduce the I4AN reference model that exposes recurring patterns; and (iv) present basic and applied patterns for reuse in PSE that aim at improving reuse efficiency and lowering risks. We evaluate the I4AN reference model and patterns with reuse scenarios

in a feasibility study in automotive manufacturing. The study results indicate that the I4AN reference model and patterns satisfy the elicited requirements and enable PSE domain experts to identify patterns for reuse and sufficiently complete sets of reusable assets in their contexts.

# Software Engineering Risks from Technical Debt in the Representation of Product/ion Knowledge

Stefan Biffl[1], Lukas Kathrein[1,3]
[1]*Inst. for Information Systems Eng.*
*Faculty of Informatics*
TU Wien, Vienna, Austria
[first].[last]@tuwien.ac.at

Arndt Lüder[2]
[2]*Inst. of Ergonomics*
*Manufacturing Sys. and Automation*
OvG U. Magdeburg, Germany
arndt.lueder@ovgu.de

K. Meixner[1,3], M. Sabou[1,3],
L. Waltersdorfer[1,3] D. Winkler[1,3]
[3]*Christian Doppler Lab SQI (sqi.at)*
TU Wien, Vienna, Austria
[first].[last]@tuwien.ac.at

***Abstract*—In the multi-disciplinary *production systems engineering* (PSE) process, software engineers depend on requirements and design rationales coming from product and production process planning, summarized as *product/ion* knowledge. Unfortunately, the engineering artifacts coming from product/ion planning often represent important product/ion knowledge incompletely and not well integrated, leading to risks regarding software engineering quality. In this paper, we report on a case study at a large industrial PSE organization, investigating *Technical Debt* (TD) effects, items, and causes in PSE process documentation and configuration management according to the VDI guideline 3695 Part 2. We focus on requirements for and issues in the representation of product/ion knowledge in the engineering data provided to software engineers. Based on data elicited from PSE domain experts, we model TD concepts based on the *Quality Function Deployment* method as foundation for TD analysis and risk management. The initial validation with domain experts revealed how software engineers could benefit from improved product/ion knowledge modeling as foundation for better understanding the rationale of engineering design decisions.**

***Keywords—Multi-Disciplinary Production Systems Engineering, Product/ion Knowledge, Product-Process-Resource (PPR) Model, Process Management, Technical Debt***

## I. Introduction

In *production systems engineering* (PSE) organizations, many different engineering disciplines work together, such as basic system and process planners, detailed automation engineers and production optimizers, for fulfilling customer requirements towards the *Industrie 4.0* vision [1] regarding, for example, production system throughput and quality. In a typical PSE process, the domain experts work in parallel in discipline-specific workgroups that exchange engineering artifacts for iterative improvement. For making informed design decisions, industrial automation and software engineers depend on the high quality of input artifacts that contain software requirements as well as results and rationale of system design decisions [4][5].

Unfortunately, the quality of *software engineering* (SE) results, such as software code governing the transportation system of a production plant, is subject to risks due to the missing or incorrect representation of product/ion knowledge, i.e., knowledge on characteristics of the product, produced by the plant or characteristics of the industrial production process and their relationships to characteristics of the production system [13].

DOI reference number: 10.18293/SEKE2019-037

An example for such a relation is a *fragile product* that imposes limitations on the maximal acceleration during the transport between production system components. If a software engineer sets the transport speed high to maximize the system throughput, the product quality may suffer, leading to the costly redesign of the overall system. Limited awareness of domain experts on the knowledge requirements of partner roles in the project may lead to insufficient descriptions of relevant engineering data and knowledge. Risks from decisions based on insufficient and often incomplete information or from unplanned effort due to unreliable communication between basic and detailed planners could be better managed with adequate knowledge representations of product/ion knowledge throughout the process. Kathrein *et al.* point in [13] out that *engineering organizations* (EOs), as defined in the VDI 3695 [28], tend to focus on discipline-specific outcomes rather than on the collaboration of domain experts. The domain experts suffer from low quality of collaboration artifacts, but do not, in general, have the knowledge or the budget to improve the collaboration process.

In this paper, we investigate *Technical Debt* (TD) in the representation of product/ion knowledge in engineering artifacts exchanged between PSE workgroups as foundation for analyzing and managing risks from TD effects, items, and causes in a PSE organization. An example for TD is a missing or incomplete engineering process description, which makes it hard to manage projects across several domains and work groups. In this paper we adapt the definition of TD by Li *et al.* [16] according to engineering artifacts and the collaboration process: *TD are violations in engineering artifacts compared to best practices of engineering process documentation and configuration for collaborative workgroups in the PSE domain*. Main goal is to identify TD throughout the engineering process, for better PSE process management, in particular, SE risks.

We report on results from a case study at a large industrial PSE organization on TD regarding process documentation according to the PSE domain VDI Guideline 3695 Part 2 [28] concerning the *procedure model for project activities* and *configuration management in an engineering organization*. We focus on eliciting requirements for and in the representation of product/ion knowledge supporting software engineers in their decision-making process. In the case study, we identified TD items, where one TD item is a unit bearing quality risk [16], on insufficient description of engineering process and information in the data exchange process and insufficient representation of product/ion knowledge. Based on collected data samples, we relate TD concepts to each other and investigate their relationships based on the *Quality Function Deployment (QFD)* [19] method. The *QFD* method allows analyzing and prioritizing customer requirements together with solution options. We use the *QFD* method for analyzing TD repayment options [19], e.g., which TD items should be addressed to reduce system design cost. Better understanding TD relationships is the foundation for advanced analyses of TD risks and TD repayment options.

The remainder of this paper is structured as follows: Section II summarizes related work on PSE, on knowledge representation in PSE, and on concepts of TD in PSE knowledge. Section III introduces the research questions and approach. Section IV reports case study findings regarding TD effects, items, and causes, and relates these TD concepts in a preliminary *QFD* style model. Section V discusses results and limitations of the research. Section VI concludes and provides an outlook on future research.

## II. Related Work

This section summarizes related work on *production systems engineering* (PSE), on knowledge representation in PSE, and on concepts of technical debt in PSE knowledge.

### A. Production Systems Engineering (PSE)

The engineering of production systems is a multi-disciplinary task involving various disciplines, such as mechanical, electrical, and software engineering [3]. The disciplines conduct a network of engineering activities where engineering decisions are taken and engineering data are created by engineers. The engineers use appropriate input data and engineering tools optimized for the discipline. One role, the automation engineering designs and implements the hardware and software of the production system control, a main software engineering task in PSE [27].

Within PSE, the importance of digital models is increasing. New activities related to the development and use of life cycle crossing digital shadows of complete production systems and their components are envisioned to enable the *Industrie 4.0* vision [17]. These models shall contain all relevant data and knowledge on production systems aspects. This includes the description of the involved production system components, the production processes they execute, and the product resulting from the production process. Schleipen *et al.* [24] calls this *PPR knowledge* and Kathrein *et al.* [13] use the term *product/ion knowledge.* In this paper, we build on the *PPR* concept to identify shortcomings regarding knowledge representation that introduce risks to SE activities.

### B. Knowledge Representation in PSE

Engineering knowledge is a specific kind of knowledge, oriented towards the production of artifacts, and, as such, requires knowledge modeling and representation approaches that differ from other types of knowledge, such as taxonomical knowledge characteristic for the life sciences domain [25]. Ontologies are information artefacts that have been used extensively to explicitly represent such engineering knowledge. This is for example investigated by Ekaputra *et al.* [7] highlighting different ontology-based data integration strategies, possible objectives an engineering organization has, as well as data source types used.

Sabou *et al.* [23] provide an overview of such ontologies and classify them in terms of the aspects of the PPR process that they cover. For example, OntoCAPE[1] [20] is an ontology for supporting *computer-aided process engineering* (CAPE) focusing on describing production process information. The *Semantic Sensor Network* (SSN)[2] ontology, developed at W3C[3], is well suited to describe process states and their observability, as well as resource states. The *Automation Ontology* (AO) captures knowledge about industrial plants and their automation systems to support engineering simulation models [21]. AO concerns mechatronic concepts to support simulation model design and integration.

The explicit modeling of PSE knowledge is characterized by the need to address recurring modeling needs specific for this domain, including: Modeling *Part-whole relations.* Legat *et al.* [14] observe that containment hierarchies are a well-accepted and frequently occurring organizational paradigm from modeling part-whole relations in (mechatronic) engineering settings. *Modeling connections between components*. Legat *et al.* [14] observe that *interface-based composition* describes the capabilities expected from an interface to enable reasoning tasks about the correctness of a system's structure.

The modeling of recurring knowledge structures can be well addressed by the reuse of *Ontology Design Patterns* (ODPs) that model best practices applicable to typical conceptualization scenarios [10]. Indeed, ODPs exist to support the conceptual modeling of (variations) of the engineering-specific modeling scenarios mentioned above. For example, modeling *Part-whole* relations can be achieved by reusing the *PartOf* ODP[4], which allows modeling part-whole relations in a transitive fashion. The *Componency* ODP[5] is a specialization of the *PartOf* ODP for modeling *part-whole* relations distinguishing between direct and indirect (i.e., transitively-assessed) parts of an object. While there are several approaches for knowledge representation in PSE they are often not used and lead thus to TD, which is addressed in this paper.

### C. Technical Debt in PSE Knowledge

Avgeriou *et al.* [2] compare *Technical Debt* (TD) to friction in mechanical devices, requiring increasingly more energy to achieve the same results as parts deteriorate. This is also true for *software engineering* (SE), as short-term gains create friction over the lifetime of a software-intensive system that require extra effort and cost to address or to repay. To deal with TD, Avgeriou *et al.* [2] propose to analyze TD repayment options and to investigate TD from different viewpoints. TD in a system consists of TD items that are measurable in an SE artifact. Li *et al.* [16] identify ten different TD types, with effects ranging from

[1] OntoCAPE: https://www.avt.rwth-aachen.de/AVT/index.php?id=730
[2] SSN Ontology: www.w3.org/2005/Incubator/ssn/ssnx/ssn.owl
[3] https://www.w3.org/
[4] PartOf ODP: http://ontologydesignpatterns.org/wiki/Submissions:PartOf
[5] Componency ODP: http://ontologydesignpatterns.org/wiki/Submissions:Componency

inconveniences to crippling whole software systems, making future maintenance costly [2]. Martini *et al.* [18] point out that large SE companies invest a quarter of the development time in managing TD to continue providing their SE functions.

Dong and Vogel-Heuser [6] draw a comparison, based on results from two case studies, between TD in PSE and in SE, as similar effects, such as short-term cost savings or lack of experience, occur in both domains. Causes of TD manifest in various dimensions, for example mechanical, electrical, or software engineering [6]. As process improvement and data exchange processes are success-critical processes in *engineering organizations* (EOs), they identify crucial TD in design and architecture, knowledge distribution and documentation [6].

Martini *et al.* [18] show how architectural TD accumulates during development in a project until reaching a crisis point that makes refactoring inevitable, increasing business value as the short-term sins are repaid adequately. Case studies by Biffl *et al.* [4][5] and Kathrein *et al.* [12][13] investigated engineering processes of EOs with a focus on the structure of collaborations between workgroups [13] and how data is exchanged [4][5]. These works represent building blocks for this paper, as they define a coherent context with basic concepts needed for TD investigations. The research highlights multiple use cases with different levels of TD, and points out missing *product/ion-aware* (PPR) knowledge as a limitation.

## III. Research Questions and Approach

This section introduces the research questions (RQs) following the *design science* method [29], and presents an illustrating use case to investigate TD in the representation of product/ion knowledge. Similarly, as in [26], we investigate TD as a form of software engineering risks, with effects and possible causes.

*RQ1: What risks to software engineering results and activities are related to technical debt in the representation of product/ion knowledge in engineering artifacts exchanged between workgroups in production systems engineering?* From the high-level RQ1, we derive the following sub-RQs.

*RQ1a: What are effects of TD related to software engineering risks in PSE?* We identify TD effects in the PSE process in interviews with domain experts. These TD effects can be defined as process management issues, i.e., deviations from the planned engineering process, and the process executed by individual domain experts. The identification of TD effects allows highlighting risks known to SE, but not to domain experts in PSE.

*RQ1b: What are TD items regarding the VDI Guideline 3695 Part 2 in engineering artifacts exchanged between workgroups in PSE?* The VDI Guideline 3695 Part 2 [28] provides valuable insights in describing engineering organizations (EOs) and potential improvement steps. The guideline provides a set of best practices that should be followed in an EO and allows analyses similar to code reviews in SE. Therefore, we define TD items in the PSE process by comparing selected target states in the VDI 3695 to the *as-is engineering process*.

*RQ1c: What are causes regarding elicited TD items?* As foundation for managing TD, we elicit in the case study candidate TD causes in the engineering organization. TD causes strengthen the deviation between the as-is and VDI 3695 defined process and are important to address TD items and SE risks.

RQ2: *How do TD concepts in the data exchange process relate to each other?* After identifying TD concepts, we model their relationships as foundation for analyzing the impact of TD causes on TD items and effects, with a focus on SE concepts.

*RQ2a: How do TD effects and TD items relate to each other?* Main outcome of this RQ is a table based on the *QFD* method [19], developed with quality managers, who are responsible for defining an ideal PSE process across all involved disciplines and for possible improvement steps. The *QFD* method facilitates prioritizing relationships between TD effects and TD items that are relevant to reduce SE risks.

*RQ2b: How do TD items and causes relate to each other?* There are many and diverse TD cause candidates that can have different impacts on TD items. This RQ investigates most relevant TD causes to influence the TD items and effects. Main outcome is a table depicting relationships of TD causes and items based on the QFD method [19], created with quality managers.

To answer the RQs, we followed a case study design [22] by adhering to the following case study plan. *[Objective]* Exploring an existing engineering process *[Case]* in a large PSE organization. *[Theory]* Following the design science cycle according to Wieringa [29] in a holistic case study, *[Goal]* we identify common concepts at collaboration interfaces between PSE workgroups, and identify information bottlenecks regarding TD effects. *[Method]* Through seven semi-structured interviews (in a funnel approach) [22], *[Selection]* we elicit representative data from domain experts and investigate TD effects, items, and causes.

According to the design science cycle [29], this paper focuses on workshops and interviews regarding TD effects, causes, items, and the types and strengths of the relationships between the TD concepts. We discuss likely causes for the TD items found. Based on Matook and Indulska [19], we adapt the QFD method to focus in this paper on two dimensions of the QFD *House of Quality* (see Section V). In cooperation with quality managers, responsible for improving the PSE process, we design tables based on the QFD method [19] for investigating TD cause candidates. Finally, we present a conceptual evaluation, discussing presented repayment options to address SE-relevant TD in the multi-disciplinary engineering process.

Kathrein *et al.* [12][13] elicited the illustrating use case in Fig. 1 for *data exchange in the PSE process*. In this paper, the use case frames the election of TD concepts in the case study. In the beginning, the *system planner* (SP) receives product specifications from the customer (1) and aims at providing a competitive offer and at deriving specific knowledge on the production system for later use. This process is similar to software architecture design [14]. Output of this step (lilac arrow) are resource documents regarding the plant layout, calculations, and assembly sequences, delivered to the *process planner* (PP) (2).

Upon receiving the artifacts, the *PP* investigates these artifacts with a common schema that domain experts have developed over decades. For example, the first column always is the module identifier followed by the module name and a reference to an existing CAD drawing if possible. Main goal is to derive basic variations of the previously offered production system for detailing mechanical aspects. However, if important product aspects and design decisions are not documented, the *PP* has to call back the *SP*, e.g., via e-mail or telephone (3). Final output of this step are detailed descriptions of the production system as foundation for production optimization and PLC *software engineering* in the form of automation tasks (4).

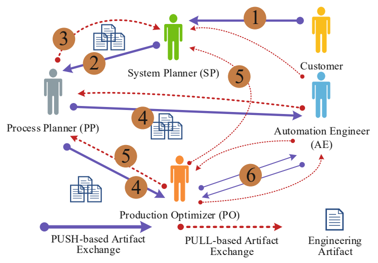


Figure 1. Use case depicting the AS-IS data exchange process.

The *production optimizer (PO)* receives all basic plans and tries to minimize the cycle times of the plant. However, this work requires different product/ion knowledge aspects and may cause many calls back to the *SP* and *PP* (5). The *PO* collaborates with the *automation engineer* (*AE*) (6), who is responsible for PLC *software engineering* tasks. From basic plans (4), the *AE* derives specific PLC software code. Goal of the *AE* is to trace design decisions as foundation for making informed design variations, such as the parameterization of the software and systems that execute production processes, e.g., the speed and acceleration of transport processes. In the next section we investigate this case study regarding TD effects, items and causes.

## IV. Case Study Results and Technical Debt Model

This section reports on findings from the case study regarding TD effects, items, and causes, and relates these TD concepts in a preliminary QFD model for the case study context.

### A. *Case Study results on TD effects (RQ1a)*

**TD effects.** Regarding the use case, data exchange process, the following TD effects came up frequently in workshop sessions.

*TD-E1 High effort for tracing design decisions.* High unplanned effort in SE activities to collect information on the rationale of design decisions to sufficiently understand what changes in the system design make sense in the production process context.

*TD-E2 Data quality risks in engineering artifacts.* Low quality of engineering artifacts may limit the production system capabilities and reduce reuse opportunities of system components.

*TD-E3 Risk of economic project failure* due to cost for unplanned effort for collecting information and due to risk of limited production system quality and capabilities.

### B. *Case Study results on TD items (RQ1b)*

The TD item description contains the following sections: *name and acronym* of the TD item; *motivation* of the typical context and short-term benefits of the TD item; *definition* of the TD item as a violation of the VDI Guideline 3695 Part 2 [28], (see Section II.A); *measurement* definition on the presence of the TD item; *relationships to effects* including long-term impact from the presence of the TD item; and hypothetical *relationships to causes*, including technical decisions or postponed best-practice activities. Based on the TD item description, we identified the following TD items.

**Engineering process description insufficient (TD1Proc)** *Motivation.* The requirements for the engineering process depend on the project and on the specific engineers conducting the engineering tasks. Therefore, engineering process models may exist on an abstract level, but do not cover engineering information exchange in sufficient detail. The domain experts focus on engineering production systems and rather than on formally defining the engineering process in detail, with the short-term benefit of starting quickly, following a method they prefer to use. The engineering process models are not maintained and often diverge from actual project practice. Similar TD concepts in SE are missing documentation of application program interfaces (APIs) and software engineering processes in general.

*Definition.* The VDI Guideline 3695 Part 2 [28], procedure model for project activities, defines the target state A as "*the staff knows the procedure model and can explain how they use it in the project.*" However, in the case study context, the domain experts referred to only experience-based informal processes, with limited awareness of the impact of actions in the process, in particular data exchange with process partners, beyond the immediate workgroup of the domain expert, such as extra effort and risk of engineering data consumers.

*Measurement.* There is no formal description of discipline-specific engineering process steps and of the collaboration processes between the disciplines involved in the engineering process. The staff does not know about their engineering process description including the impact of exchanged information and the required data maturity.

*Relationships to effects. TD-E1, TD-E3* (see Table I in Section IV.C). TD symptoms include high effort for communication and rework due to shortcomings in data exchange, in particular for SE, as the SE activities depend in inputs from several disciplines that may be incomplete or even contradicting (see Fig. 1).

*Relationships to causes*. See Table II, in Section IV.C. In the case study, main causes came from insufficient means that hinder the description of the engineering process.

**Information description insufficient (TD2Inf)** (in exchanged engineering data)**.** *Motivation.* In the case study context, the domain experts focus only on the data relevant to their own discipline and do not consider dependencies to related disciplines. They often use tool-specific data exports, such as component lists or CAD drawings, and Excel as a general-purpose information exchange artifact. Short-term benefits include saving effort for the data provider and flexible choice of means for the provider when collecting the engineering data to exchange.

Similar TD concepts in SE are missing documentation of interfaces, code, and implementation details.

*Definition.* The VDI Guideline 3695 Part 2 [28], configuration management, defines the target state A as *"... there are discipline-specific procedures for configuration identification, configuration monitoring, [...] Within a discipline, all employees follow common guidelines.*" However, in the case study, domain experts found artifacts not to be managed, but simply to evolve over time according to engineering personnel experience, without specific consideration for dependencies between engineering artifacts in different disciplines, which poses risks for SE activities that depend on consistent and complete inputs (see Fig. 1).

*Measurement.* There is no formal description of engineering artifacts and data, including dependencies between engineering disciplines, such as product, process, and system design. There is no configuration history for backtracking design decisions.

*Relationships to effects.* See Table I, in Section IV.C. TD symptoms include high effort and risk for propagating changes to systems design across disciplines, in particular for SE, when receiving inputs from several engineering disciplines.

*Relationships to causes*. See Table II, in Section IV.C.

**Product/ion (PPR) knowledge representation insufficient (TD3PPR)** (in exchanged engineering data)**.** *Motivation.* Domain experts in production process design have product/ion (PPR) knowledge that would be, in many cases, important to engineers in later stages of PSE and optimization, in particular, for SE activities. However, the process designer tends to provide her engineering partners with hard-coded production system parameters rather than PPR knowledge as there is no dedicated tool or modeling language to allow the effective and efficient representation of PPR knowledge. Short-term benefit for the process designer is saving effort for modeling the PPR knowledge. Similar TD concept in SE would be missing information on non-functional requirements for a software system.

*Definition.* The VDI Guideline 3695 Part 2 [28], configuration management, defines the target state D as "*system-assisted cross-discipline*" configuration management to enable "*consistency check [...] at an early stage*". However, without sufficient PPR knowledge representation, consistency checks between production process design and production system design are difficult, error-prone, and take considerable expert effort.

*Measurement.* The engineering data model misses representations for expressing PPR knowledge and rationale to trace design decisions, such as production system temperature settings to the welding temperature and force of a metal joining process.

*Relationships to effects.* See Table I, in Section IV.C. TD symptoms include in SE activities considerable costs of errors from changes and effort for preventing defects after changes.

*Relationships to causes*. See Table II, in Section IV.C. In the case study context, main cause candidates include workgroup-specific optimization of the engineering organization and insufficient means to express PPR knowledge.

### *C. Case Study results on TD cause candidates (RQ1c)*

**Cause candidates linked to context in the engineering organization**, often for economic and historic reasons in the EO.

*A1. Workgroup-related profit centers* lead to the local optimization of workgroups with limited concerns for the optimization of projects across workgroups, often at the expense of SE.

*A2. Engineering habit trained by discipline-specific education* leads to engineers focusing on good results in their workgroup. Engineers are, in general, not aware about work tasks, dependencies, and problems in other workgroups, unless a partner asks them for an improvement.

*A3. Unclear responsibilities of domain experts in data exchange process* lead to ad-hoc procedures and data definitions.

*A4. Limited collaboration effort across work groups* without a dedicated role for coordinating the work across workgroups.

**Cause candidates from engineering process description**

*B1. Engineering process modelled as an artifact-based workflow, not as a data-related workflow* makes it hard to describe dependencies between SE and other disciplines, such as consistency rules that relate to the data model, not to artifacts.

*B2. Engineering process defined, but not useful.* There is a workflow definition for a process. However, the definition may be abstract and lack important description of content dependencies, such as relationships between the product and resource design, tainting the usefulness of the definition.

*B3. Engineering process defined, but not operational.* There is a process description. However, missing technical foundations, such as adequate process description concepts or tool support, make it hard or impossible to conduct the process.

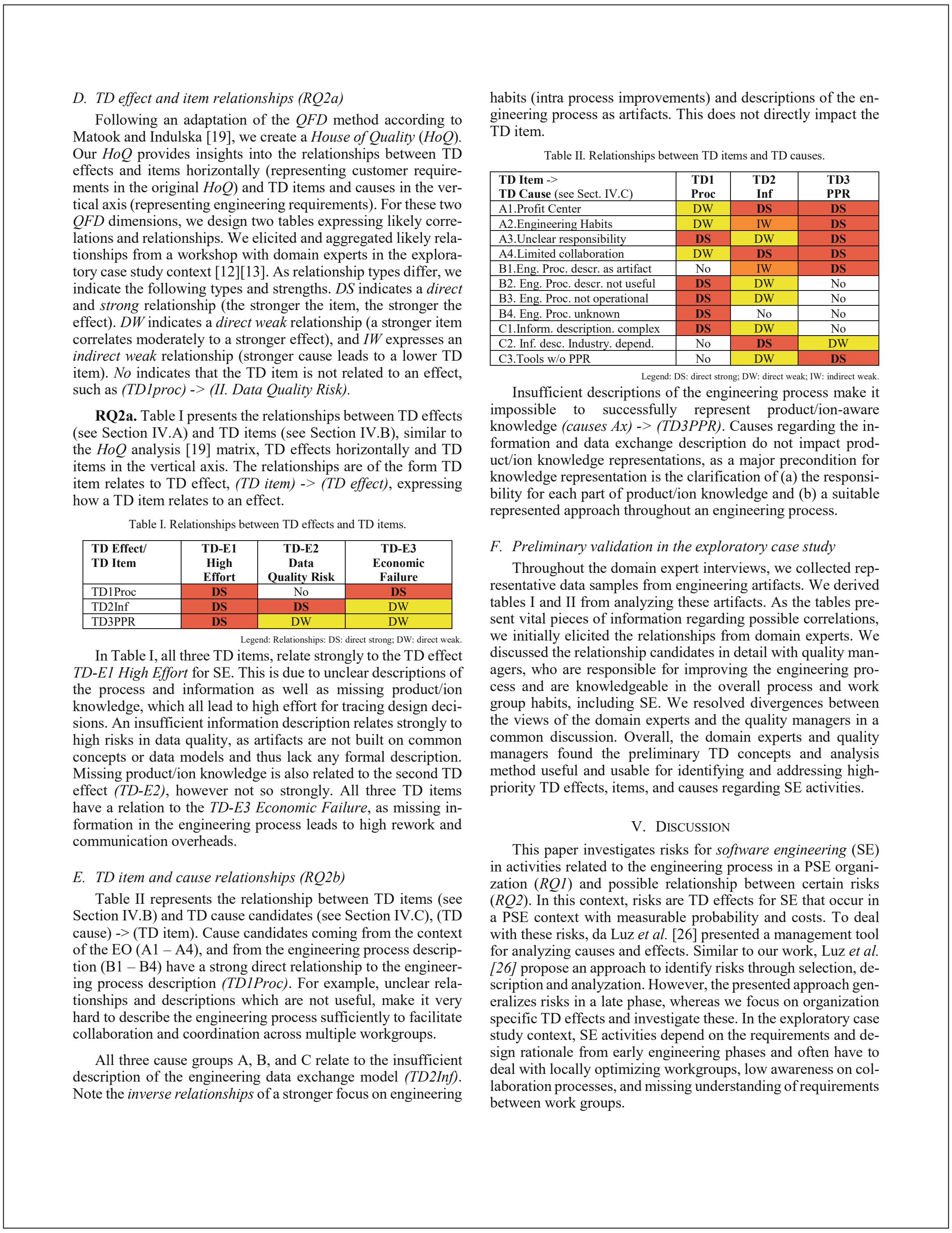
*B4. Engineering process defined, but not known to stakeholders.* There is a process description somewhere in a manual. However, the relevant actors in the project are not aware of the process description for their daily work.

**Cause candidates linked to information description**

*C1. Description of complex dependencies required* due to a large number of disciplines (often 15 or more) in a PSE project. Complex descriptions of processes and artifacts and their dependencies in an *engineering organization* (EO) lead to a very complex network (consider Fig. 1, scaled up).

*C2. Industry-dependent information description.* The description of information depends on the industry and has to be adapted accordingly. There is no general standard that could be applied directly. There is no general EO model as the industry domains require a variety of EO structures and behavior

*C3. Tool-driven process without product/ion (PPR) information description.* Often, the process is defined based on a specific tool chain. Therefore, the functional and data export capabilities of the tool determine the exchanged information. The process is not aware of PPR as the discipline-specific tools only know the PPR knowledge that is relevant within the discipline.



### D. *TD effect and item relationships (RQ2a)*

Following an adaptation of the *QFD* method according to Matook and Indulska [19], we create a *House of Quality* (*HoQ*). Our *HoQ* provides insights into the relationships between TD effects and items horizontally (representing customer requirements in the original *HoQ*) and TD items and causes in the vertical axis (representing engineering requirements). For these two *QFD* dimensions, we design two tables expressing likely correlations and relationships. We elicited and aggregated likely relationships from a workshop with domain experts in the exploratory case study context [12][13]. As relationship types differ, we indicate the following types and strengths. *DS* indicates a *direct* and *strong* relationship (the stronger the item, the stronger the effect). *DW* indicates a *direct weak* relationship (a stronger item correlates moderately to a stronger effect), and *IW* expresses an *indirect weak* relationship (stronger cause leads to a lower TD item). *No* indicates that the TD item is not related to an effect, such as *(TD1proc) -> (II. Data Quality Risk).*

**RQ2a.** Table I presents the relationships between TD effects (see Section IV.A) and TD items (see Section IV.B), similar to the *HoQ* analysis [19] matrix, TD effects horizontally and TD items in the vertical axis. The relationships are of the form TD item relates to TD effect, *(TD item) -> (TD effect)*, expressing how a TD item relates to an effect.

Table I. Relationships between TD effects and TD items.

| **TD Effect/ TD Item** | **TD-E1 High Effort** | **TD-E2 Data Quality Risk** | **TD-E3 Economic Failure** |
|---|---|---|---|
| TD1Proc | **DS** | No | **DS** |
| TD2Inf | **DS** | **DS** | DW |
| TD3PPR | **DS** | DW | DW |

Legend: Relationships: DS: direct strong; DW: direct weak.

In Table I, all three TD items, relate strongly to the TD effect *TD-E1 High Effort* for SE. This is due to unclear descriptions of the process and information as well as missing product/ion knowledge, which all lead to high effort for tracing design decisions. An insufficient information description relates strongly to high risks in data quality, as artifacts are not built on common concepts or data models and thus lack any formal description. Missing product/ion knowledge is also related to the second TD effect *(TD-E2)*, however not so strongly. All three TD items have a relation to the *TD-E3 Economic Failure*, as missing information in the engineering process leads to high rework and communication overheads.

### E. *TD item and cause relationships (RQ2b)*

Table II represents the relationship between TD items (see Section IV.B) and TD cause candidates (see Section IV.C), (TD cause) -> (TD item). Cause candidates coming from the context of the EO (A1 – A4), and from the engineering process description (B1 – B4) have a strong direct relationship to the engineering process description *(TD1Proc)*. For example, unclear relationships and descriptions which are not useful, make it very hard to describe the engineering process sufficiently to facilitate collaboration and coordination across multiple workgroups.

All three cause groups A, B, and C relate to the insufficient description of the engineering data exchange model *(TD2Inf)*. Note the *inverse relationships* of a stronger focus on engineering habits (intra process improvements) and descriptions of the engineering process as artifacts. This does not directly impact the TD item.

Table II. Relationships between TD items and TD causes.

| **TD Item ->** **TD Cause** (see Sect. IV.C) | **TD1 Proc** | **TD2 Inf** | **TD3 PPR** |
|---|---|---|---|
| A1.Profit Center | DW | **DS** | **DS** |
| A2.Engineering Habits | DW | IW | **DS** |
| A3.Unclear responsibility | **DS** | DW | **DS** |
| A4.Limited collaboration | DW | **DS** | **DS** |
| B1.Eng. Proc. descr. as artifact | No | IW | **DS** |
| B2. Eng. Proc. descr. not useful | **DS** | DW | No |
| B3. Eng. Proc. not operational | **DS** | DW | No |
| B4. Eng. Proc. unknown | **DS** | No | No |
| C1.Inform. description. complex | **DS** | DW | No |
| C2. Inf. desc. Industry. depend. | No | **DS** | DW |
| C3.Tools w/o PPR | No | DW | **DS** |

Legend: DS: direct strong; DW: direct weak; IW: indirect weak.

Insufficient descriptions of the engineering process make it impossible to successfully represent product/ion-aware knowledge *(causes Ax) -> (TD3PPR)*. Causes regarding the information and data exchange description do not impact product/ion knowledge representations, as a major precondition for knowledge representation is the clarification of (a) the responsibility for each part of product/ion knowledge and (b) a suitable represented approach throughout an engineering process.

### F. *Preliminary validation in the exploratory case study*

Throughout the domain expert interviews, we collected representative data samples from engineering artifacts. We derived tables I and II from analyzing these artifacts. As the tables present vital pieces of information regarding possible correlations, we initially elicited the relationships from domain experts. We discussed the relationship candidates in detail with quality managers, who are responsible for improving the engineering process and are knowledgeable in the overall process and work group habits, including SE. We resolved divergences between the views of the domain experts and the quality managers in a common discussion. Overall, the domain experts and quality managers found the preliminary TD concepts and analysis method useful and usable for identifying and addressing high-priority TD effects, items, and causes regarding SE activities.

## V. Discussion

This paper investigates risks for *software engineering* (SE) in activities related to the engineering process in a PSE organization (*RQ1*) and possible relationship between certain risks (*RQ2*). In this context, risks are TD effects for SE that occur in a PSE context with measurable probability and costs. To deal with these risks, da Luz *et al.* [26] presented a management tool for analyzing causes and effects. Similar to our work, Luz *et al. [26]* propose an approach to identify risks through selection, description and analyzation. However, the presented approach generalizes risks in a late phase, whereas we focus on organization specific TD effects and investigate these. In the exploratory case study context, SE activities depend on the requirements and design rationale from early engineering phases and often have to deal with locally optimizing workgroups, low awareness on collaboration processes, and missing understanding of requirements between work groups.

For software engineers, the *high effort* comes from risks regarding rework efforts due to frequent and late changes coming from earlier phases. As software engineers highly depend on weakly documented design decisions from early phases, a repayment option for TD is better knowledge representation of product/ion knowledge throughout the engineering process. The *low data exchange quality* impacts software engineers, who are made responsible for low quality system output, even if they write high quality code, but based on weakly communicated early design decisions. TD repayment for reducing the SE risk should focus on improving the documentation and communication of design decisions that are directly related to high-quality SE results. Finally, issues regarding unplanned efforts for reworks in the software design due to low system quality decisions may exceed the budget available to SE, leading to local *economic failure.*

The VDI Guideline 3695 Part 2 [28] was used to investigate TD items (*RQ1b*). This guideline can be seen similar to best practices for SE code development, and our analysis is equivalent to a code review. An *unclear engineering process description* makes it hard for software engineers to reliably configure production systems, as input from several disciplines may be contradicting. The missing collaboration in PSE forces software engineers to take the risky decision on which inputs to consider or ignore. Further, the *information description is not sufficient*. This makes it unclear for software engineers where to look for reliable information, as data syntax and semantics may change frequently, making it hard to validate input data and to automate the data exchange process. Software engineers thus often work based on risky assumptions. Finally, *missing product/ion aware knowledge* makes it hard in SE to take informed decisions for adapting the software system design if the preferred system design option is not feasible.

In *RQ1c*, we investigated possible causes regarding elicited TD items. Causes linked to the *context of production systems engineering* cannot be directly influenced by SE actors, but require the insight of PSE managers. We discussed the preliminary results at the case study EO with quality managers, who found the analysis useful for considering and prioritizing improvement options. Cause candidates linked to the *engineering process description* clearly motivate the need for better means of PPR knowledge representation as a foundation for process descriptions, considering conceptual, language, and usability aspects. This is similar to the need for proper software architecture descriptions identified by Guessi *et al.* [11]. The last group we identified are cause candidates linked to the *information description*. For example, it is challenging to combine methods for data integration [7] with domain-specific standards, such as *AutomationML* [7] or ontologies [23].

A *repayment option* to address TD items related to weak collaboration of workgroups is a new role, the *data curator*, similarly as presented in [9]. This role would be responsible for consolidating a common data exchange model and describe the engineering process adequately. This new role should needs to understand the requirements and limitations of the involved workgroups, in particular SE activities. As TD effects, items and causes are related to each other, we used the *Quality Function Deployment* [19] method to investigate relationships between TD effects, items and causes (*RQ2*).

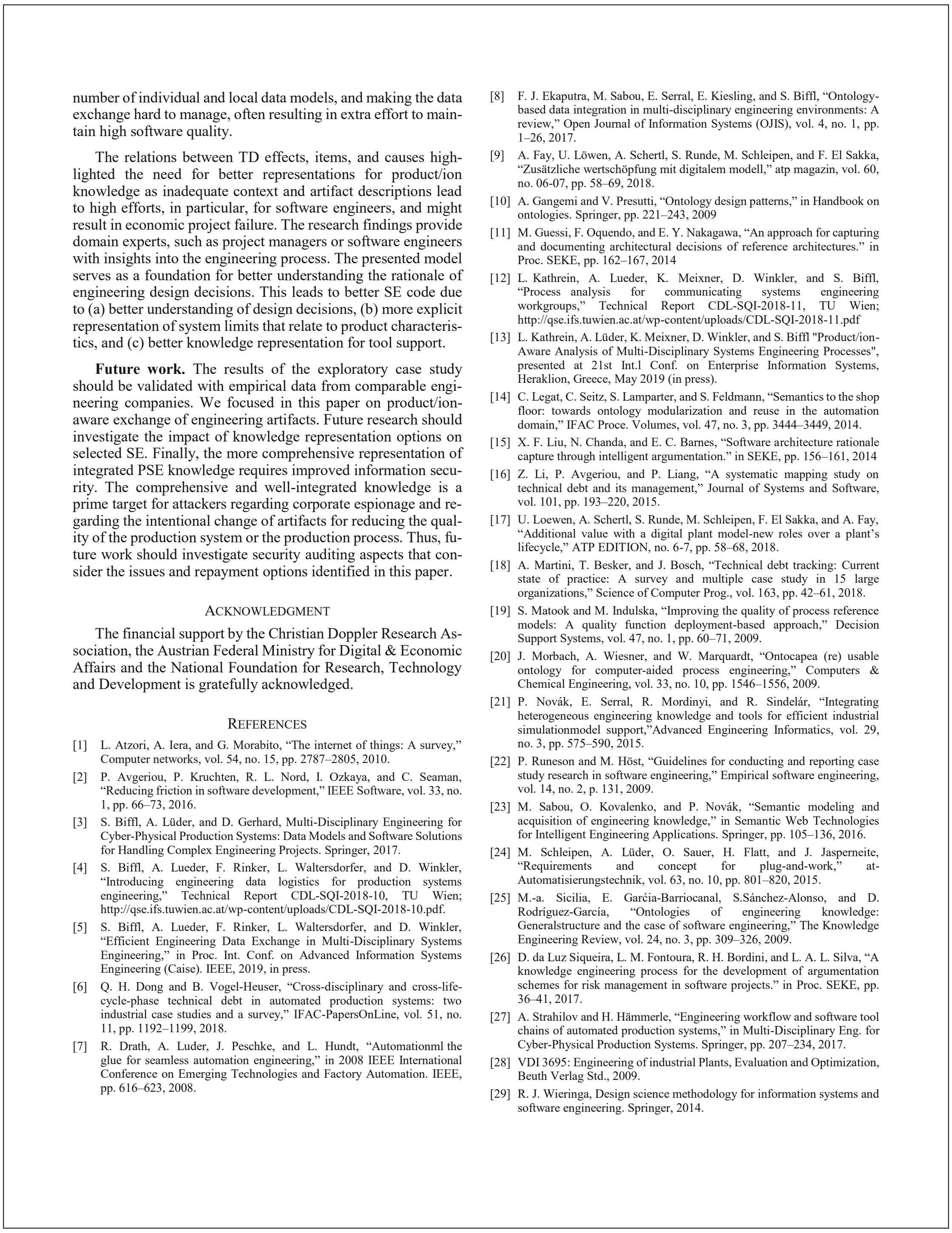
In *RQ2a* we investigated how TD effects and TD items relate to each other. The *TD-E1 High Effort* is strongly and directly connected to all three TD items. This is obvious as reworks are often needed to compensate missing descriptions or information bottlenecks where especially software engineers are affected. Interesting is, that the *TD-E2 Data Quality Risk* is only weakly connected to *PPR* knowledge representation, even though this is an important information exchange concept. For SE this means that design decisions from early phases do not impact the code development so much as the overall information description, this could be for example the selection of an easily changeable component in the user interface.

*RQ2b* investigated how TD items and causes are related. Nearly all causes regarding the information description strongly impact the process description. This clearly motivates the need for better knowledge representation approaches, as the current engineering process is either not described, or the description is not useful or unknown to domain experts. As there are currently no tools that support the expression of PPR knowledge, software engineers could address this open issue to improve product/ion-aware knowledge representation and further allow a backflow of SE knowledge into early engineering phases as foundation for designing better reusable system parts.

**Limitations.** The research of this paper followed a case study in an engineering organization. However, the case study focused on only one company, which may not be representative for all EOs in general. While the domain experts in the study were very knowledgeable, their number was limited due to resource limitations of the available experts besides their daily business obligations. We found that the engineering process description may highly depend on the context, domain, and organization, thus future case studies should consider these variation points. While the domain experts found the preliminary list of TD effects, items, and causes, and their relationships useful for reflecting on TD repayment options in the case study context, these results require validation and discussion on comparable context for strengthening the external validity of the results.

## VI. Conclusion and Future Work

In engineering organizations, software engineers join the PSE process in a late phase and are concerned with detailing SE aspects of the software-intensive system. However, software engineers often only receive poorly described design decisions in form of engineering artifacts making it hard for them to derive adequate new (software) engineering knowledge or tracing the earlier design decisions. These shortcomings lead to risks regarding the SE quality and impact the project effort negatively, endangering project success. In this paper, we reported on a case study at a large industrial engineering organization with the focus of investigating technical debt (TD) effects, items, and causes as risks for software engineers. Results highlight that TD can slow down engineering organizations, making it hard to manage processes where multiple domains are involved. Main insight for addressing the found challenges is the introduction of a new role, the *data curator*, to facilitate the collaboration across workgroups. The results highlight requirements for the representation of product/ion knowledge in the engineering data provided to software engineers. Engineering data is heterogeneous and there are no guidelines for basic planners, leading to a large



number of individual and local data models, and making the data exchange hard to manage, often resulting in extra effort to maintain high software quality.

The relations between TD effects, items, and causes highlighted the need for better representations for product/ion knowledge as inadequate context and artifact descriptions lead to high efforts, in particular, for software engineers, and might result in economic project failure. The research findings provide domain experts, such as project managers or software engineers with insights into the engineering process. The presented model serves as a foundation for better understanding the rationale of engineering design decisions. This leads to better SE code due to (a) better understanding of design decisions, (b) more explicit representation of system limits that relate to product characteristics, and (c) better knowledge representation for tool support.

**Future work.** The results of the exploratory case study should be validated with empirical data from comparable engineering companies. We focused in this paper on product/ion-aware exchange of engineering artifacts. Future research should investigate the impact of knowledge representation options on selected SE. Finally, the more comprehensive representation of integrated PSE knowledge requires improved information security. The comprehensive and well-integrated knowledge is a prime target for attackers regarding corporate espionage and regarding the intentional change of artifacts for reducing the quality of the production system or the production process. Thus, future work should investigate security auditing aspects that consider the issues and repayment options identified in this paper.

## Acknowledgment

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital & Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

## References

[1] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” Computer networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] P. Avgeriou, P. Kruchten, R. L. Nord, I. Ozkaya, and C. Seaman, “Reducing friction in software development,” IEEE Software, vol. 33, no. 1, pp. 66–73, 2016.

[3] S. Biffl, A. Lüder, and D. Gerhard, Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects. Springer, 2017.

[4] S. Biffl, A. Lueder, F. Rinker, L. Waltersdorfer, and D. Winkler, “Introducing engineering data logistics for production systems engineering,” Technical Report CDL-SQI-2018-10, TU Wien; http://qse.ifs.tuwien.ac.at/wp-content/uploads/CDL-SQI-2018-10.pdf.

[5] S. Biffl, A. Lueder, F. Rinker, L. Waltersdorfer, and D. Winkler, “Efficient Engineering Data Exchange in Multi-Disciplinary Systems Engineering,” in Proc. Int. Conf. on Advanced Information Systems Engineering (Caise). IEEE, 2019, in press.

[6] Q. H. Dong and B. Vogel-Heuser, “Cross-disciplinary and cross-life-cycle-phase technical debt in automated production systems: two industrial case studies and a survey,” IFAC-PapersOnLine, vol. 51, no. 11, pp. 1192–1199, 2018.

[7] R. Drath, A. Luder, J. Peschke, and L. Hundt, “Automationml the glue for seamless automation engineering,” in 2008 IEEE International Conference on Emerging Technologies and Factory Automation. IEEE, pp. 616–623, 2008.

[8] F. J. Ekaputra, M. Sabou, E. Serral, E. Kiesling, and S. Biffl, “Ontology-based data integration in multi-disciplinary engineering environments: A review,” Open Journal of Information Systems (OJIS), vol. 4, no. 1, pp. 1–26, 2017.

[9] A. Fay, U. Löwen, A. Schertl, S. Runde, M. Schleipen, and F. El Sakka, “Zusätzliche wertschöpfung mit digitalem modell,” atp magazin, vol. 60, no. 06-07, pp. 58–69, 2018.

[10] A. Gangemi and V. Presutti, “Ontology design patterns,” in Handbook on ontologies. Springer, pp. 221–243, 2009

[11] M. Guessi, F. Oquendo, and E. Y. Nakagawa, “An approach for capturing and documenting architectural decisions of reference architectures.” in Proc. SEKE, pp. 162–167, 2014

[12] L. Kathrein, A. Lueder, K. Meixner, D. Winkler, and S. Biffl, “Process analysis for communicating systems engineering workgroups,” Technical Report CDL-SQI-2018-11, TU Wien; http://qse.ifs.tuwien.ac.at/wp-content/uploads/CDL-SQI-2018-11.pdf

[13] L. Kathrein, A. Lüder, K. Meixner, D. Winkler, and S. Biffl "Product/ion-Aware Analysis of Multi-Disciplinary Systems Engineering Processes", presented at 21st Int.l Conf. on Enterprise Information Systems, Heraklion, Greece, May 2019 (in press).

[14] C. Legat, C. Seitz, S. Lamparter, and S. Feldmann, “Semantics to the shop floor: towards ontology modularization and reuse in the automation domain,” IFAC Proce. Volumes, vol. 47, no. 3, pp. 3444–3449, 2014.

[15] X. F. Liu, N. Chanda, and E. C. Barnes, “Software architecture rationale capture through intelligent argumentation.” in SEKE, pp. 156–161, 2014

[16] Z. Li, P. Avgeriou, and P. Liang, “A systematic mapping study on technical debt and its management,” Journal of Systems and Software, vol. 101, pp. 193–220, 2015.

[17] U. Loewen, A. Schertl, S. Runde, M. Schleipen, F. El Sakka, and A. Fay, “Additional value with a digital plant model-new roles over a plant’s lifecycle,” ATP EDITION, no. 6-7, pp. 58–68, 2018.

[18] A. Martini, T. Besker, and J. Bosch, “Technical debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations,” Science of Computer Prog., vol. 163, pp. 42–61, 2018.

[19] S. Matook and M. Indulska, “Improving the quality of process reference models: A quality function deployment-based approach,” Decision Support Systems, vol. 47, no. 1, pp. 60–71, 2009.

[20] J. Morbach, A. Wiesner, and W. Marquardt, “Ontocapea (re) usable ontology for computer-aided process engineering,” Computers & Chemical Engineering, vol. 33, no. 10, pp. 1546–1556, 2009.

[21] P. Novák, E. Serral, R. Mordinyi, and R. Sindelár, “Integrating heterogeneous engineering knowledge and tools for efficient industrial simulationmodel support,”Advanced Engineering Informatics, vol. 29, no. 3, pp. 575–590, 2015.

[22] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” Empirical software engineering, vol. 14, no. 2, p. 131, 2009.

[23] M. Sabou, O. Kovalenko, and P. Novák, “Semantic modeling and acquisition of engineering knowledge,” in Semantic Web Technologies for Intelligent Engineering Applications. Springer, pp. 105–136, 2016.

[24] M. Schleipen, A. Lüder, O. Sauer, H. Flatt, and J. Jasperneite, “Requirements and concept for plug-and-work,” at-Automatisierungstechnik, vol. 63, no. 10, pp. 801–820, 2015.

[25] M.-a. Sicilia, E. Garćia-Barriocanal, S.Sánchez-Alonso, and D. Rodríguez-García, “Ontologies of engineering knowledge: Generalstructure and the case of software engineering,” The Knowledge Engineering Review, vol. 24, no. 3, pp. 309–326, 2009.

[26] D. da Luz Siqueira, L. M. Fontoura, R. H. Bordini, and L. A. L. Silva, “A knowledge engineering process for the development of argumentation schemes for risk management in software projects.” in Proc. SEKE, pp. 36–41, 2017.

[27] A. Strahilov and H. Hämmerle, “Engineering workflow and software tool chains of automated production systems,” in Multi-Disciplinary Eng. for Cyber-Physical Production Systems. Springer, pp. 207–234, 2017.

[28] VDI 3695: Engineering of industrial Plants, Evaluation and Optimization, Beuth Verlag Std., 2009.

[29] R. J. Wieringa, Design science methodology for information systems and software engineering. Springer, 2014.

### 5.2.3 Organizing Reuse with Capabilities and Skills

#### Citation

[135] **K. Meixner**, F. Rinker, L. Waltersdorfer, A. Lueder, and S. Biffl. Organizing reuse for production systems engineering with capabilities and skills. *At-Automatisierungstechnik*, 71(2):116–127, 2023. doi: 10.1515/AUTO-2022-0120

#### Aim

The publication "Organizing reuse for production systems engineering with capabilities and skills" aims at improving the reuse of engineering assets by decoupling production process and resource models. The publication provides (i) requirements for capability and skill reuse, (ii) a framework to define the adaptation of VDI 3695 activities to support domain and application engineering for capabilities and skills, and (iii) how the framework can facilitate reuse along the production system life-cycle.

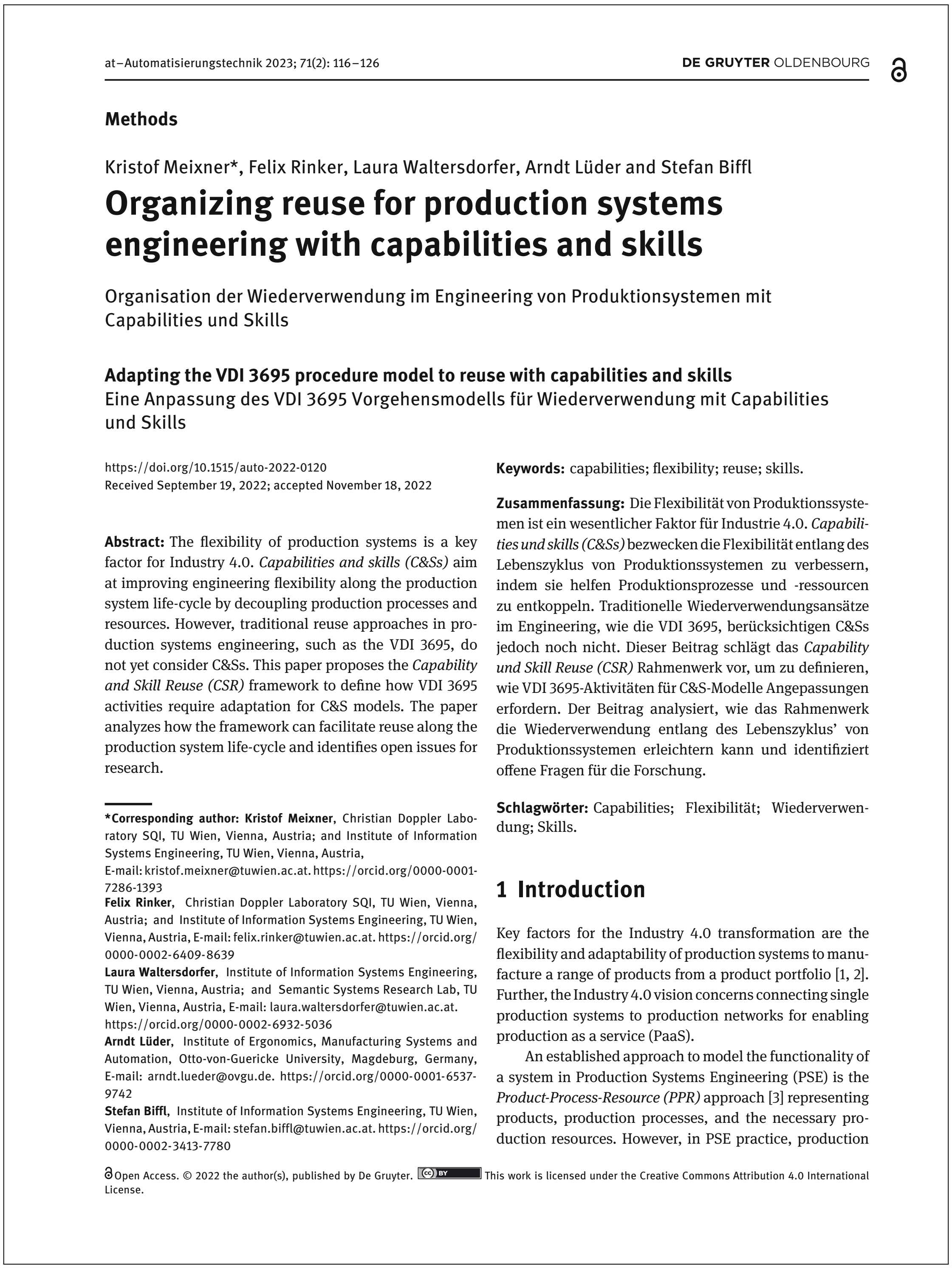
#### Contribution to the thesis

This publication contributes requirements for capability and skill reuse, and to the research goals CPPS reuse method for capabilities and skills (**G3**., **G2**.).

This publication contributes to **RQ2**. and **RQ3**. by addressing the VDI 3695 measures of *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *enhanced quality and consistency* C4., *facilitation of interdisciplinary collaboration* C5..

#### Abstract

The flexibility of production systems is a key factor for Industry 4.0. Capabilities and skills (C&Ss) aim at improving engineering flexibility along the production system life-cycle by decoupling production processes and resources. However, traditional reuse approaches in production systems engineering, such as the VDI 3695, do not yet consider C&Ss. This paper proposes the Capability and Skill Reuse (CSR) framework to define how VDI 3695 activities require adaptation for C&S models. The paper analyzes how the framework can facilitate reuse along the production system life-cycle and identifies open issues for research.



at – Automatisierungstechnik 2023; 71(2): 116–126 DE GRUYTER OLDENBOURG

**Methods**

Kristof Meixner*, Felix Rinker, Laura Waltersdorfer, Arndt Lüder and Stefan Biffl

# Organizing reuse for production systems engineering with capabilities and skills

Organisation der Wiederverwendung im Engineering von Produktionsystemen mit Capabilities und Skills

**Adapting the VDI 3695 procedure model to reuse with capabilities and skills**
Eine Anpassung des VDI 3695 Vorgehensmodells für Wiederverwendung mit Capabilities und Skills

https://doi.org/10.1515/auto-2022-0120
Received September 19, 2022; accepted November 18, 2022

**Abstract:** The flexibility of production systems is a key factor for Industry 4.0. *Capabilities and skills (C&Ss)* aim at improving engineering flexibility along the production system life-cycle by decoupling production processes and resources. However, traditional reuse approaches in production systems engineering, such as the VDI 3695, do not yet consider C&Ss. This paper proposes the *Capability and Skill Reuse (CSR)* framework to define how VDI 3695 activities require adaptation for C&S models. The paper analyzes how the framework can facilitate reuse along the production system life-cycle and identifies open issues for research.

***Corresponding author: Kristof Meixner**, Christian Doppler Laboratory SQI, TU Wien, Vienna, Austria; and Institute of Information Systems Engineering, TU Wien, Vienna, Austria, E-mail: kristof.meixner@tuwien.ac.at. https://orcid.org/0000-0001-7286-1393
**Felix Rinker**, Christian Doppler Laboratory SQI, TU Wien, Vienna, Austria; and Institute of Information Systems Engineering, TU Wien, Vienna, Austria, E-mail: felix.rinker@tuwien.ac.at. https://orcid.org/0000-0002-6409-8639
**Laura Waltersdorfer**, Institute of Information Systems Engineering, TU Wien, Vienna, Austria; and Semantic Systems Research Lab, TU Wien, Vienna, Austria, E-mail: laura.waltersdorfer@tuwien.ac.at. https://orcid.org/0000-0002-6932-5036
**Arndt Lüder**, Institute of Ergonomics, Manufacturing Systems and Automation, Otto-von-Guericke University, Magdeburg, Germany, E-mail: arndt.lueder@ovgu.de. https://orcid.org/0000-0001-6537-9742
**Stefan Biffl**, Institute of Information Systems Engineering, TU Wien, Vienna, Austria, E-mail: stefan.biffl@tuwien.ac.at. https://orcid.org/0000-0002-3413-7780

**Keywords:** capabilities; flexibility; reuse; skills.

**Zusammenfassung:** Die Flexibilität von Produktionssystemen ist ein wesentlicher Faktor für Industrie 4.0. *Capabilities und skills (C&Ss)* bezwecken die Flexibilität entlang des Lebenszyklus von Produktionssystemen zu verbessern, indem sie helfen Produktionsprozesse und -ressourcen zu entkoppeln. Traditionelle Wiederverwendungsansätze im Engineering, wie die VDI 3695, berücksichtigen C&Ss jedoch noch nicht. Dieser Beitrag schlägt das *Capability und Skill Reuse (CSR)* Rahmenwerk vor, um zu definieren, wie VDI 3695-Aktivitäten für C&S-Modelle Angepassungen erfordern. Der Beitrag analysiert, wie das Rahmenwerk die Wiederverwendung entlang des Lebenszyklus' von Produktionssystemen erleichtern kann und identifiziert offene Fragen für die Forschung.

**Schlagwörter:** Capabilities; Flexibilität; Wiederverwendung; Skills.

## 1 Introduction

Key factors for the Industry 4.0 transformation are the flexibility and adaptability of production systems to manufacture a range of products from a product portfolio [1, 2]. Further, the Industry 4.0 vision concerns connecting single production systems to production networks for enabling production as a service (PaaS).

An established approach to model the functionality of a system in Production Systems Engineering (PSE) is the *Product-Process-Resource (PPR)* approach [3] representing products, production processes, and the necessary production resources. However, in PSE practice, production processes and resources, and their models, are often tightly coupled, impeding production flexibility. Complementing the PPR approach, *capabilities and skills (C&Ss)* aim to support flexibility along the production system life-cycle by decoupling and abstracting production processes and their requirements from the production resources that execute these processes [4]. For instance, recent research proposed formal and machine-readable C&S descriptions, e.g., using ontologies, to overcome tacit expert knowledge [5, 6]. Such semantic descriptions also aim at an automated matching of production process requirements to resources, easing PSE and lowering manual work that is prone to error [7]. Further works investigated the automated derivation of capability descriptions along with executable skills [8]. Machine-readable capability and skill (C&S) descriptions combined with automated matching and decoupled executable production services should facilitate the required flexibility and creation of production networks.

A crucial prerequisite for efficient and high-quality PSE and operation is the systematic reuse of engineering artifacts, such as resources with their models and reference sheets. Systematic reuse promises to reduce engineering cost and time to market, lower maintenance effort and improve the quality of products [9]. Jazdi et al. [10] investigated the *project-independent activities (PIAs)* of VDI 3695 [11] to improve the efficiency of PSE by increasing artifact reusability. The same applies to C&Ss, where *reusability* and *reuse* were identified in a recent literature survey as both a requirement for and a benefit of using C&Ss [4]. This especially holds in distributed environments, such as PaaS.

However, to the best of our knowledge, the elicitation of C&Ss from existing engineering artifacts, such as PPR models, for their systematic reuse has not been reported. Therefore, we aim in this paper to address the research question: *How do VDI 3695 activities for domain and application engineering require adaptation to facilitate reuse with capabilities and skills in PSE?*

To address this research question in this work, we provide the following contributions. We categorize requirements towards C&Ss from a recent literature survey [4] and investigate how these requirements need to be considered to facilitate C&S-based reuse. We propose the *Capability and Skill Reuse (CSR) framework* for the elicitation of C&Ss from engineering models and artifacts and their reuse. Therefore, we describe how the VDI 3695 reuse activities for domain and application engineering in PSE require adaptation to enable C&S-based reuse. Further, we discuss the benefits and limitations of C&S models as a foundation to facilitate reuse along the production system life-cycle.

The remainder of the paper is structured as follows. Section 2 describes the background and related work on knowledge representation and reuse in PSE. Section 3 categorizes requirements towards C&S and introduces the Capability and Skill Reuse (CSR) framework for the reuse of production system models and artifacts. Sections 4 and 5 discuss the research results and conclude.

## 2 Background

This section summarizes the background and related work and sketches an illustrative use case.

### 2.1 Knowledge representation in PSE

PSE consists of several life-cycle phases, from basic and detailed planning to commissioning and operation. Additionally, PSE takes place in a multidisciplinary environment, where engineers from various domains, such as mechanical or electrical engineering, maintain different views on the production system [12]. The PPR approach [3] unifies three main aspects of PSE. The model represents input and output ***p**roducts*, production ***p**rocesses* required to transform input into output products, and production ***r**esources* that automate the production processes. The Formal Process Description (FPD), defined in the VDI 3682 [13], provides a visual and formal model to describe these aspects.

Pfrommer et al. [14] introduced *skills* as a complementary element to PPR. They defined *skills* as semantic, vendor-independent representations of process functionality *required by a product* and *provided by a resource*. Keddis et al. [15] distinguished *required capabilities*, defined in production step plans, from *provided capabilities*, implemented by resources. While a recent literature survey found early literature on C&S to use the concepts interchangeably [4], later works distinguished more clearly between C&S. Nevertheless, both concepts aim to abstract and decouple processes and resources for more flexible production.

Several works proposed models for C&S, including their relations to PPR, and formal machine-readable specification, e.g., via ontologies [5, 16, 17]. Järvenpää et al. [7] investigated how formally described resource capabilities can be automatically matched to product requirements. Furthermore, recent research studied the automated derivation and execution of skills, e.g., via OPC UA [8, 18].

This paper builds on the PPR and C&S concepts, in particular, on the definitions and the model of C&S described in [19]. In their model, capabilities are implementation-independent specifications of functions in industrial production. While production resources can provide such capabilities, production processes can require them, encapsulating the product requirements. On the other hand, skills are executable implementations of capabilities on a resource (cf. Figure 1), where several skills might realize a capability.

## 2.2 Knowledge reuse in PSE

Jazdi et al. [10] investigated how PSE efficiency can be increased by identifying reusable engineering artifacts and systematically using them extensively in upcoming engineering projects. Their work is based on the VDI 3695 procedure model for project activities [11] and the *two-life-cycle model* from Software Product Line Engineering (SPLE) [9].

The VDI 3695 [11] for optimizing engineering organizations describes a *two-phase procedure model*. The model consists of *project-independent activities (PIAs)* to identify and provide reusable artifacts and *project-dependent activities (PDAs)* that use these artifacts (cf. Figure 2). The PIAs for domain engineering consist of (i) an *analysis* of the domain and artifacts suited for reuse, (ii) the *planning* of a reference architecture and reusable artifacts, and (iii) the *realization* and *testing* of reusable artifacts. Jazdi et al. [10] translate these activities into detailed tasks for PSE. The VDI 3695 specifies five target states of reuse maturity in PSE that range from isolated reuse by single engineers to reuse with reference models and standards. The *PDAs* for application engineering consist of (i) the *acquisition* and requirements engineering of new PSE projects, (ii) the *planning* and *realization* of a production system or production services using the reference architecture and reusable artifacts, and (iii) *commissioning* of the realized system and services. Ideally, requirements, acquired knowledge, and

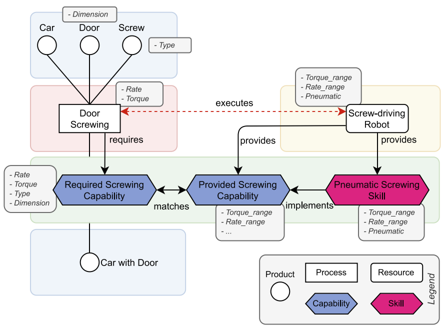


**Figure 1:** PPR model for a *screwing process* in VDI 3682 [13] notation *without* (dashed connection *executes*) and *with* C&Ss.

artifacts from the individual projects are inputs to the PIAs (cf. Figure 2, dashed arrow).

Similarly, *SPLE* investigates the reuse, flexibility, and configuration of software portfolios and their engineering [20]. Therefore, van der Linden et al. [9] identified the four fundamental principles of *variability management*, *business-centric* and *architecture-centric* engineering, and the *two-life-cycle* approach. The *two-life-cycle* approach describes *domain engineering* and *application engineering*, similar to PIAs and PDAs of the VDI 3695 [11]. The reusable artifacts are stored in a *common artifact repository* (cf. Figure 2) for use in the PDAs. SPLE also investigated models and methods to represent and manage variability in artifacts and their configuration. Two SPLE approaches to model and configure variability, also mentioned by Jazdi et al. [10], are *feature modeling* and *decision modeling* [20].

A recent survey [4] elicited expected *requirements* towards C&S and *benefits* of C&S in PSE. The survey reported *reusability* both as a requirement to enable C&S-based PSE and as a benefit as C&Ss foster *reuse* of knowledge. However, to the best of our knowledge, very little work on the systematic reuse of C&S in PSE has been reported. Therefore, this paper investigates the reuse in and for C&S-based PSE, in particular, the elicitation and abstraction of C&Ss from engineering artifacts, such as PPR models.

## 2.3 Illustrative use case

To illustrate the CSR framework, this section reports on a use case abstracted from real-world applications. These applications stem from engineering organizations of high-performance automation for car part manufacturing in Germany and Austria [21]. In particular, we consider *joining processes* of large car parts, such as doors, to car bodies.

In the use case, we consider a system integrator which, for particular customers, plans and engineers work line production systems that manufacture a portfolio of car parts automated in typical car production plants. The plant operators should be able to offer their production services in a marketplace in the future, aiming at PaaS. Therefore, the system integrator wants to reuse existing well-described engineering artifacts, such as robot cell models, for various company-wide projects. To this end, the solution candidates should be decoupled from specific product and production process requirements of previous projects.

Figure 1 shows a section of a PPR model in VDI 3682 notation [13] with the products (represented as circles in a blue frame), e.g., the door, one process step (depicted

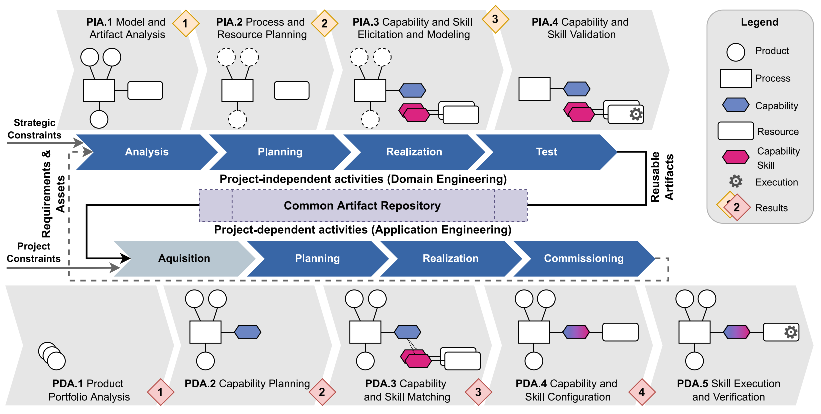

PIA.1 Model and Artifact Analysis
1
PIA.2 Process and Resource Planning
2
PIA.3 Capability and Skill Elicitation and Modeling
3
PIA.4 Capability and Skill Validation
Legend
Product
Process
Capability
Resource
Capability Skill
Execution
2
Results
Strategic Constraints
Analysis
Planning
Realization
Test
Requirements & Assets
Project-independent activities (Domain Engineering)
Common Artifact Repository
Reusable Artifacts
Project-dependent activities (Application Engineering)
Project Constraints
Aquisition
Planning
Realization
Commissioning
PDA.1 Product Portfolio Analysis
1
PDA.2 Capability Planning
2
PDA.3 Capability and Skill Matching
3
PDA.4 Capability and Skill Configuration
4
PDA.5 Skill Execution and Verification

**Figure 2:** The *Capability and Skill Reuse (CSR) framework* for PPR models and artifacts, based on the *VDI 3695* guideline [11].

as a rectangle in a red frame), i.e., the door screwing process, and a resource (shown as a rounded rectangle in a yellow frame), i.e., the screw-driving robot. A corresponding engineering artifact can be, e.g., the model in the Product-Process-Resource Domain-Specific Language (PPR–DSL) [22].

In the use case, the door is mounted to a car body with screws. The screws have different types depending on the doors, which have different dimensions. The processes have technical and economic requirements, such as the required torque or the production rate. Furthermore, the processes need to consider the characteristics of the products, i.e., the torque and dimension. The screw-driving robot has several characteristics, such as torque or rate range, required for correct process execution. While some of these characteristics are similar in the assembly line, others are variants of originating processes and resources.

Traditional PSE and reuse [10, 23] often maintain a quite resource-centered perspective, where processes are modeled only as part of the resource and its behavior (cf. dashed red arrow in Figure 1).

In C&S-based PSE, the process requirements are modeled as *required capabilities*, e.g., a *screwing capability* (cf. Figure 1) or even more abstract a *joining capability*. However, some characteristics, such as electric or pneumatic screwing, might be irrelevant to the production processes. Resources, such as screw-driving robots, on the other hand, provide the means to execute a functionality, in this case, to join two parts, respectively, screw them together. In C&S-based PSE, the functionalities of resources are modeled as *provided capabilities* and implemented as, e.g., *skills* [19]. This way, the required and provided capabilities can be matched, e.g., in the engineering phase or on a marketplace.

## 3 The CSR framework for reuse with capabilities and skills

This section categorizes requirements for C&Ss regarding their relevance for reuse and presents the *Capability and Skill Reuse (CSR)* framework.

### 3.1 Requirements towards capability- and skill-based engineering and reuse

Froschauer et al. [4] elicited requirements towards C&Ss in a literature survey. We categorize these requirements to identify which of them are particularly relevant in what reuse activity of the CSR framework. In Table 1, we introduce four categories and assign each requirement to one of them.[1] In the next section, we use this categorization to highlight the relevance of the particular requirements in the activities of the CSR framework.

The first category concerns the *description of capabilities* to support their interpretation by machines and humans, exchange, and ideally openness [24]. Therefore, capability descriptions shall be *formal* and *vendor-neutral*.

1 Note that the requirement *reusability* in [4] is not part of this categorization due to the overall focus of this paper on reuse.

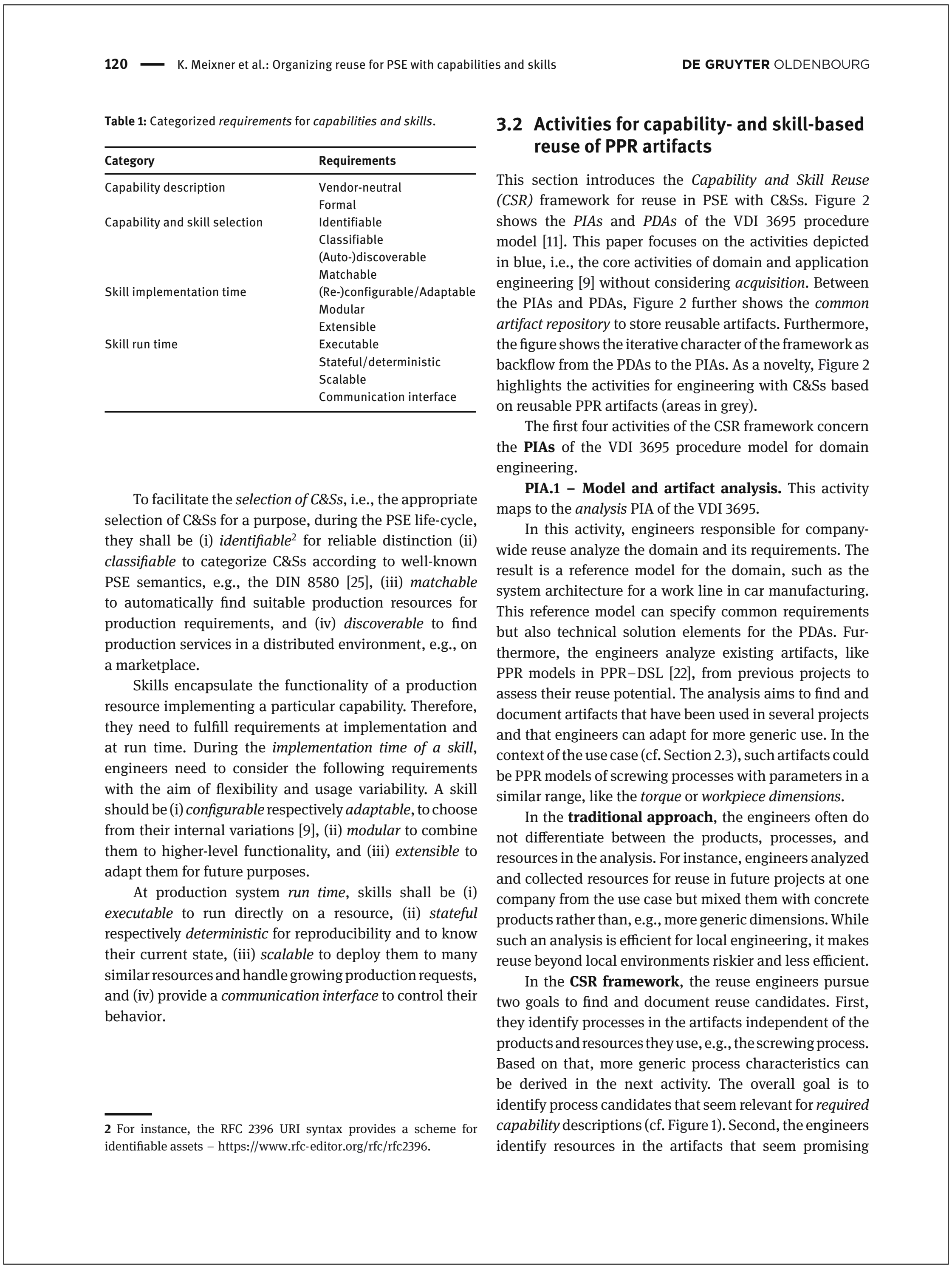


**Table 1:** Categorized *requirements* for *capabilities and skills*.

| Category | Requirements |
|---|---|
| Capability description | Vendor-neutral |
| | Formal |
| Capability and skill selection | Identifiable |
| | Classifiable |
| | (Auto-)discoverable |
| | Matchable |
| Skill implementation time | (Re-)configurable/Adaptable |
| | Modular |
| | Extensible |
| Skill run time | Executable |
| | Stateful/deterministic |
| | Scalable |
| | Communication interface |

To facilitate the *selection of C&Ss*, i.e., the appropriate selection of C&Ss for a purpose, during the PSE life-cycle, they shall be (i) *identifiable*[2] for reliable distinction (ii) *classifiable* to categorize C&Ss according to well-known PSE semantics, e.g., the DIN 8580 [25], (iii) *matchable* to automatically find suitable production resources for production requirements, and (iv) *discoverable* to find production services in a distributed environment, e.g., on a marketplace.

Skills encapsulate the functionality of a production resource implementing a particular capability. Therefore, they need to fulfill requirements at implementation and at run time. During the *implementation time of a skill*, engineers need to consider the following requirements with the aim of flexibility and usage variability. A skill should be (i) *configurable* respectively *adaptable*, to choose from their internal variations [9], (ii) *modular* to combine them to higher-level functionality, and (iii) *extensible* to adapt them for future purposes.

At production system *run time*, skills shall be (i) *executable* to run directly on a resource, (ii) *stateful* respectively *deterministic* for reproducibility and to know their current state, (iii) *scalable* to deploy them to many similar resources and handle growing production requests, and (iv) provide a *communication interface* to control their behavior.

2 For instance, the RFC 2396 URI syntax provides a scheme for identifiable assets – https://www.rfc-editor.org/rfc/rfc2396.

## 3.2 Activities for capability- and skill-based reuse of PPR artifacts

This section introduces the *Capability and Skill Reuse (CSR)* framework for reuse in PSE with C&Ss. Figure 2 shows the *PIAs* and *PDAs* of the VDI 3695 procedure model [11]. This paper focuses on the activities depicted in blue, i.e., the core activities of domain and application engineering [9] without considering *acquisition*. Between the PIAs and PDAs, Figure 2 further shows the *common artifact repository* to store reusable artifacts. Furthermore, the figure shows the iterative character of the framework as backflow from the PDAs to the PIAs. As a novelty, Figure 2 highlights the activities for engineering with C&Ss based on reusable PPR artifacts (areas in grey).

The first four activities of the CSR framework concern the **PIAs** of the VDI 3695 procedure model for domain engineering.

**PIA.1 – Model and artifact analysis.** This activity maps to the *analysis* PIA of the VDI 3695.

In this activity, engineers responsible for company-wide reuse analyze the domain and its requirements. The result is a reference model for the domain, such as the system architecture for a work line in car manufacturing. This reference model can specify common requirements but also technical solution elements for the PDAs. Furthermore, the engineers analyze existing artifacts, like PPR models in PPR–DSL [22], from previous projects to assess their reuse potential. The analysis aims to find and document artifacts that have been used in several projects and that engineers can adapt for more generic use. In the context of the use case (cf. Section 2.3), such artifacts could be PPR models of screwing processes with parameters in a similar range, like the *torque* or *workpiece dimensions*.

In the **traditional approach**, the engineers often do not differentiate between the products, processes, and resources in the analysis. For instance, engineers analyzed and collected resources for reuse in future projects at one company from the use case but mixed them with concrete products rather than, e.g., more generic dimensions. While such an analysis is efficient for local engineering, it makes reuse beyond local environments riskier and less efficient.

In the **CSR framework**, the reuse engineers pursue two goals to find and document reuse candidates. First, they identify processes in the artifacts independent of the products and resources they use, e.g., the screwing process. Based on that, more generic process characteristics can be derived in the next activity. The overall goal is to identify process candidates that seem relevant for *required capability* descriptions (cf. Figure 1). Second, the engineers identify resources in the artifacts that seem promising for reuse, e.g., the screw-driving robot. For instance, one company in our use case maintains a database of regularly used resources with particular characteristics, such as power consumption. In another company, a domain analysis showed that the types of screw-driving robots could be reduced by grouping them by their functionality [21]. The overall goal is to identify functionality candidates that seem relevant as input to describe the functionalities of these resources as *provided capabilities* and implement them as *skills*.

The **result of the analysis** (cf. Figure 2 yellow diamond 1) is a *reference model* for the domain, e.g., for joining work lines, and a *documentation* of (partial) PPR artifacts identified as processes or resources. This reference model and the documentation are the inputs for activity PIA.2.

**PIA.2 – Process and resource planning.** This activity maps to the *planning* PIA of the VDI 3695.

In this activity, the engineers define the utilization of the assets from the analyzed PPR artifacts, e.g., usage as reference assets or parameterized assets in future projects. Therefore, the engineers need to generalize and categorize the assets and plan their parameterization, i.e., assess their variability and configuration options.

In the use case, we consider the PPR model of the door screwing process. First, the PPR artifacts might need to be split to single out the section with the screwing process. Then, the engineers need to generalize the products as the products are most likely different in other projects. For instance, the type of screw used, e.g., with a specific inventory number, needs to be generalized to its relevant properties, e.g., slotted or Phillips head. Similarly, the door might be generalized to a workpiece with particular dimensions. Figure 1 illustrates this generalization of products as dashed outlines in activity PIA.2.

In the **traditional approach**, process and resource assets are often mixed or merged in the artifacts. For instance, the screw-driving robot might be modeled to directly manipulate the products, only implicitly representing a screwing process. Thus, artifacts resulting from this planning activity are less modular and decoupled and, hence, represent the requiring and provisioning side of production insufficiently.

In the **CSR approach**, the engineers decide which (parts of the) artifacts concern process capabilities, i.e., required, or resource capabilities, i.e., provided. If an artifact mixes process requirements and resource functionality, the engineers plan how to separate the process from the resource descriptions. In the use case, the PPR artifact requires splitting or remodeling to separate the process and the resource, e.g., into different artifacts that can then be imported as libraries. The engineers must then categorize and group the processes and resources by similar requirements and functionality. This task can follow domain-specific guidelines, such as the DIN 8580[3] for manufacturing methods. For instance, screwing and welding processes can be part of a higher-level group of joining processes. Similarly, catalogs, such as company-specific taxonomies or databases or the *EClass* standard,[4] can help to categorize resources.

The **result of the planning activity** (cf. Figure 2 yellow diamond 2) are *separated and categorized processes (with generalized products) and resources*. These resulting artifacts are the input for activity PIA.3.

**PIA.3 – Capability and skill elicitation and modelling.** This activity maps to the PIAs *planning* and *realization* of the VDI 3695.

In this activity, domain engineers realize the reusable assets as PPR artifacts. For instance, they create templates from particular assets that can be parameterized and reused in different PSE projects.

In the **traditional approach**, such PPR template designs often consist of resources only that realize the functionality for the production of a product. Therefore, the resulting artifacts concern product and process requirements as well as the functionality of particular resources like robots. While this might make the configuration of such templates easier, it makes, e.g., the exchange of resources solely based on the required functionality harder [7].

In the **CSR framework**, the domain engineers elicit and model processes and their requirements as process capabilities. In several cases, the engineers might want to aggregate the found process requirements for an abstraction to higher-level groups, e.g., serving a range of parameter values [26]. Similarly, the engineers model the resource functionality as capabilities and potentially implement them as skills. Modeling the C&Ss requires using appropriate models or languages, such as ontologies [5, 17] or domain-specific languages [22]. For the skill implementations, engineers can utilize technologies, such as OPC UA [8, 18] or PackML.[5]

In the use case, we would model the screwing process as required capability, e.g., in a process ontology [5], with the required rate, torque, and screw type as configurable parameters. The screw-driving robot would be modeled

3 DIN 8580 – https://standards.globalspec.com/std/1742169/DIN%208580.
4 EClass – https://eclass.eu/en/eclass-standard.
5 PackML – https://www.omac.org/packml.

as provided capability with its rate and torque range, i.e., the minimal and maximal values, e.g., 100 Nm to 200 Nm. Furthermore, the concrete functionality of the screw-driving robot would be implemented as a skill, e.g., with specific control software, that implements the provided capability.

The capabilities shall fulfill the *capability description* and *C&S selection* requirements (cf. Table 1) that are then relevant in the PDAs. Similarly, the skill implementations shall fulfill the *implementation time* requirements. While this is an extra effort in this step, fulfilling these requirements pays off in the project-dependent realization.

The **result of the realization** is modeled as C&Ss that are deployed into the common artifact repository using an agreed-on structure. Figure 2 illustrates these artifacts as processes with generalized products with their corresponding capabilities in blue and as resources with their corresponding capabilities (and skills) in magenta.

**PIA.4 – Capability and skill validation.** This activity maps to the *testing* PIA of the VDI 3695.

After modeling the capabilities and implementing the skills, the provided C&Ss must be tested and validated to qualify them for the production systems. This task should already be executed on physical resources, such as a testbed, indicated by the cog wheel in the resources.

Therefore, skills must fulfill the *run time* requirements. This means, they need to be *executable*, *stateful* and *deterministic*, and *scalable* [4]. Furthermore, they need to have a *communication interface* to control their behavior.

The artifacts resulting from the PIAs are stored in the common artifact repository for use in the PDAs.

The activities in the lower part of Figure 2 concern the project-dependent activity (PDA) of the VDI 3695 procedure model.

**PDA.1 – Product portfolio analysis.** This activity maps to the *planning* PDA of the VDI 3695 and is similar in the **traditional approach** and the **CSR framework**.

In this activity, application engineers analyze the product portfolio that the production system should manufacture. Therefore, they investigate its production requirements, such as quality concerns or the planned order quantity, based on input from the *acquisition* activity. Figure 2 shows a product portfolio as small product icons. For instance, in the use case, a range of different doors types shall be mounted to similar types of car bodies.

In the use case, the engineers analyze the doors and car bodies with their commonalities and variability. From this analysis, they can derive which products are similar enough to manufacture them on one production system, e.g., door type *A* and *B* with similar dimensions in a robot cell.

The **result of this analysis** shall be a *bill of materials* for the products with *corresponding variability and configuration models* [9]. In Figure 2 the red label 1 at the lower part represents the results that are input to PDA.2.

**PDA.2 – Capability planning.** This activity maps to the *planning* and *realization* PDAs of the VDI 3695.

In this activity, application engineers receive the production system requirements and the product portfolio analysis from PDA.1. From this data and the reference architecture from PIA.1, they derive a suitable production system architecture. Second, the application engineers define the requirements and the functionality for the products' single production steps. Therefore, they take up the analyzed product portfolio, investigate ways to assemble these products, and derive properties for the production steps. In the use case, the engineers decide, e.g., to mount the doors to the car body with screws of a certain type using a screwing process with a specific torque and rate.

In the **traditional approach**, the engineers often define a bill of processes and tie them to reusable resource templates from the common artifact repository. However, without adequate abstraction, this limits the exchange and reconfiguration of resources and impedes the search for adequate production services, e.g., in a production network.

In the **CSR framework**, the engineers take up the product portfolio description to plan the capabilities required to produce the products. In the use case, the engineers define that a door is screwed to the car body, e.g., with a Phillips and 150 Nm torque. Based on these requirements, the engineers search in the common artifact repository for process capabilities that fit their requirements, i.e., a screwing capability with configuration parameters for the screw head and the torque. Therefore, the capabilities have to fulfill the requirement categories *capability description* and *C&S selection* (cf. Table 1) to find suitable reusable capabilities in the common artifact repository.

In the second step, the engineers configure the retrieved capabilities with the known values of the products from the product portfolio. In the use case, this means configuring a screwing capability with, e.g., the required torque of 150 Nm ±5 Nm, and the required Phillips screw head. Modeling classes of parameter ranges can be helpful to support better matching to provided C&Ss in the next step.

Figure 2 illustrates the **resulting configured capability** as a PPR model with concrete products and a configured process capability in blue. These configured capabilities (cf. red diamond 2 in Figure 2) are then inputs to PDA.3.

**PDA.3 – Capability and skill matching.** This activity maps to the *planning* and *realization* PDAs of the VDI 3695.

In this activity, the product requirements, expressed through process descriptions, must be matched and bound to concrete production resources. The resources must then be configured to execute the processes correctly.

In the **traditional approach**, the engineers manually match the process requirements to resources from the common artifact repository. This step is mainly based on implicit knowledge by a single key engineer. A major limitation of this approach is the need for experts with high domain knowledge suitable to conduct this task. Thus, this task is challenging, error-prone, hard to teach, and limited by the availability of these key experts.

In the **CSR framework** application engineers use computational support to match the partially configured process capabilities with resource capabilities. Therefore, the tooling environment shall enable to match C&S descriptions. The selection requires C&Ss to be (cf. Table 1) *identifiable* to find unique ones, *classifiable* to search for them systematically, *discoverable* to retrieve them from the common artifact repository, and *matchable* to assign them to the required process capabilities. Ontology reasoning is a promising approach to match C&Ss [7]. In the use case, we would set up, e.g., an ontology query that searches for provided capabilities with a torque range that includes the configured torque of 150 Nm. The result would be the previously described provided capability with a torque range from 100 Nm to 200 Nm. Similarly, other provide capabilities with torque ranges around 150 Nm could be returned. Furthermore, the engineers can now select from different skills that implement these capabilities, e.g., based on robots that they regularly use in their projects.

The **results of this step** are matched required and provided capabilities (cf. red label 3 Figure 2) that are input to *PDA.4*. Figure 2 depicts the icon of the processes with concrete products and their required capabilities that are matched to one or more capabilities.

**PDA.4 – Capability and skill configuration.** This activity maps to the *planning* and *realization* PDAs of the VDI 3695.

In the **traditional approach**, one of two cases applies. If the artifacts are tightly coupled, they require adaption to the particular project. If the artifacts are loosely coupled, they require adaption to each other and the project. This specific adaptation makes it hard to exchange production resources in case they do not fit as expected.

In the **CSR framework**, the engineers receive C&Ss from activity PDA.3 that match each other but are only partially configured. For instance, the torque in the screwing capability has been set to a particular value, in the use case 150 Nm. However, the engineer might need to set further C&S parameters to concrete values for the production. Furthermore, the engineers need to bind the particular capability to the finally selected and configured skill.

In Figure 2 the **bound and configured capability and skill** for a process are shown in a PPR model with connected C&Ss. The results of this step (cf. red label 4 Figure 2) are input to *PDA.5*.

**PDA.5 – Skill execution.** This activity maps to the *commissioning* PDA of the VDI 3695.

During commissioning, the designed production system runs for the first time with different resource configurations for various products from the product portfolio.

In the **traditional approach**, the engineers configured the resources based on product and process knowledge which is often diluted over the engineering process. While they might have used reusable artifacts for the configuration, we argue that it is harder to reconfigure resources in case they do not prove as efficient or suitable as expected.

In the **CSR approach**, the finally configured skills are executed for the first time on the production system with the particular configuration. This activity shall ensure, beyond other things, that the C&Ss are configured correctly. To run on the production system, skills must fulfill the same requirements as in *PIA.3*, where the skills are tested and validated. Concretely, they have to fulfill the *run time* requirements of Table 1. Issues during the execution of the skills shall be fed back directly to the PIAs, e.g., validation and testing.

Experiences and assets from the PDAs are fed back to the PIAs to improve the reuse lifecycle.

As a summary, the CSR framework defines activities that motivate which tasks should be taken to (i) elicit C&Ss from previous projects, (ii) model and validate them as reusable artifacts in a common artifact repository, and (iii) use and configure them in particular engineering projects. In the next section, we discuss the CSR framework in context to the related work and research question.

## 4 Discussion

In PSE, engineering organizations aim at establishing reuse to improve the quality and efficiency of engineering [11].

This paper investigated the research question: How do VDI 3695 activities for domain and application engineering require adaptation to facilitate reuse with capabilities and skills in PSE?

To address this research question, this paper introduced the *Capability and Skill Reuse (CSR)* framework that defines how the VDI 3695 procedure model for reuse for domain and application engineering requires adaptation to support but also benefit from C&S models.

The CSR framework organizes the design and application of C&S models that fulfill essential C&S requirements (cf. Table 1) to facilitate C&S-based reuse in PSE. In particular, the CSR framework guides engineering activities to move from traditional reuse of tightly coupled engineering artifacts for a specific reference architecture towards an approach aimed at higher reuse. C&S-based reuse of loosely coupled C&S artifacts facilitates both internal reuse and the exchange of C&S on a marketplace with a wider audience, possibly for similar but different reference architectures.

In these contexts, Table 2 lists expected benefits of C&S-based (reuse in) PSE towards flexible production [4]. However, these benefits require investment into C&S-based assets, which should be organized in increments that each provide a benefit to justify the cost and mitigate migration risks.

The activities of CSR framework themselves require the means and methods of the C&S community. This includes C&S description methods, such as ontologies [5, 17] and domain-specific languages [22], and technologies to implement skills, such as OPC UA [8, 18] or PackML/ISA 88. To this end, the CSR framework requires a future discussion and validation by the community.

The explicit modeling of C&S knowledge represents currently implicit domain expert knowledge to improve the automation of designing C&S-based solutions that match defined process capability requirements. To this end, the CSR framework facilitates integrating scattered domain knowledge on reuse from several engineering disciplines,

**Table 2:** Expected *benefits* of *capabilities and skills* [4].

| Category | Benefits |
|---|---|
| Usage advancement | Flexibility |
| | Automatic matching |
| | Optimization |
| | Interoperability |
| Development streamlining | Planning efficiency |
| | Development efficiency |
| Reuse support | Abstraction |
| | C&S reuse |

in particular, product and process design, as well as a variety of detail engineering disciplines [27].

The research in this paper goes beyond the state of the art in PSE reuse [10, 11] and C&S-based engineering (i) by defining how engineers responsible for reuse can create and exchange reusable C&S-based engineering artifacts and models from PPR artifacts and (ii) by illustrating its applicability in a reuse use case.

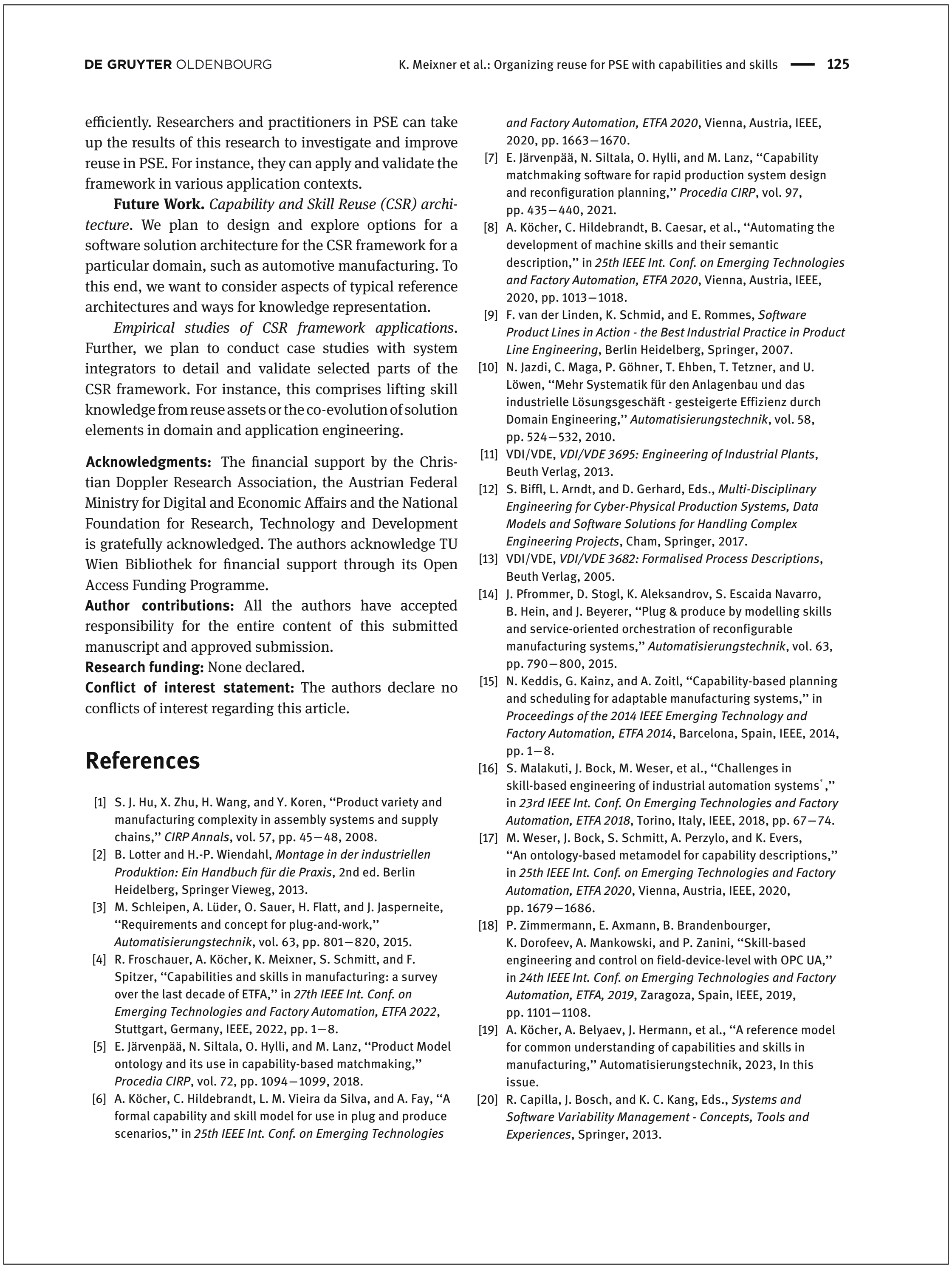
The following **limitation** requires further investigation. While there are promising contributions towards C&S-based reuse, experiments and case studies in typical application contexts are required to provide sound *empirical evidence* on the expected benefits and associated costs and risks.

## 5 Conclusion and outlook

The Industry 4.0 initiative indicated the flexibility and adaptability of systems as a crucial success factor for future production. The Product-Process-Resource (PPR) concept aims to provide a model to represent the key aspects of Production Systems Engineering (PSE). Complemented by capabilities and skills (C&Ss), which aim to abstract process requirements and resource functionality and decouple them using semantic descriptions, this approach supports the required flexibility for building production networks [4].

There is maturing work on C&S foundations for engineering and exchange in a marketplace [28]. However, less emphasis has been put on the question of how to combine reuse concerns with C&S-based engineering to provide a framework for starting and growing C&S-based engineering in a company. An example are system integrators that consider providing or procuring solution elements on a marketplace for C&Ss. However, traditional reuse approaches in PSE, such as the VDI 3695 guideline [11] and domain engineering [9, 10], do not consider C&S. Therefore, this paper introduced the Capability and Skill Reuse (CSR) framework to extend the reuse activities of the VDI 3695 procedure model towards the use of C&Ss. Therefore, we build on and integrate basic C&S representation and processing capabilities [5, 8, 17, 18].

Traditional reuse works well for a system integrator in a limited domain with well-known solution partners and a stable set of reference architecture and solution technologies. However, considering incremental investment into C&Ss seems advisable if more flexibility regarding these concerns is required. In this context, C&S-based reuse can facilitate the work of domain experts with computer functions for reuse processes to use scarce expert resources



efficiently. Researchers and practitioners in PSE can take up the results of this research to investigate and improve reuse in PSE. For instance, they can apply and validate the framework in various application contexts.

**Future Work.** *Capability and Skill Reuse (CSR) architecture*. We plan to design and explore options for a software solution architecture for the CSR framework for a particular domain, such as automotive manufacturing. To this end, we want to consider aspects of typical reference architectures and ways for knowledge representation.

*Empirical studies of CSR framework applications*. Further, we plan to conduct case studies with system integrators to detail and validate selected parts of the CSR framework. For instance, this comprises lifting skill knowledge from reuse assets or the co-evolution of solution elements in domain and application engineering.

**Acknowledgments:** The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme.
**Author contributions:** All the authors have accepted responsibility for the entire content of this submitted manuscript and approved submission.
**Research funding:** None declared.
**Conflict of interest statement:** The authors declare no conflicts of interest regarding this article.

## References

[1] S. J. Hu, X. Zhu, H. Wang, and Y. Koren, "Product variety and manufacturing complexity in assembly systems and supply chains," *CIRP Annals*, vol. 57, pp. 45–48, 2008.
[2] B. Lotter and H.-P. Wiendahl, *Montage in der industriellen Produktion: Ein Handbuch für die Praxis*, 2nd ed. Berlin Heidelberg, Springer Vieweg, 2013.
[3] M. Schleipen, A. Lüder, O. Sauer, H. Flatt, and J. Jasperneite, "Requirements and concept for plug-and-work," *Automatisierungstechnik*, vol. 63, pp. 801–820, 2015.
[4] R. Froschauer, A. Köcher, K. Meixner, S. Schmitt, and F. Spitzer, "Capabilities and skills in manufacturing: a survey over the last decade of ETFA," in *27th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA 2022*, Stuttgart, Germany, IEEE, 2022, pp. 1–8.
[5] E. Järvenpää, N. Siltala, O. Hylli, and M. Lanz, "Product Model ontology and its use in capability-based matchmaking," *Procedia CIRP*, vol. 72, pp. 1094–1099, 2018.
[6] A. Köcher, C. Hildebrandt, L. M. Vieira da Silva, and A. Fay, "A formal capability and skill model for use in plug and produce scenarios," in *25th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA 2020*, Vienna, Austria, IEEE, 2020, pp. 1663–1670.
[7] E. Järvenpää, N. Siltala, O. Hylli, and M. Lanz, "Capability matchmaking software for rapid production system design and reconfiguration planning," *Procedia CIRP*, vol. 97, pp. 435–440, 2021.
[8] A. Köcher, C. Hildebrandt, B. Caesar, et al., "Automating the development of machine skills and their semantic description," in *25th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA 2020*, Vienna, Austria, IEEE, 2020, pp. 1013–1018.
[9] F. van der Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action - the Best Industrial Practice in Product Line Engineering*, Berlin Heidelberg, Springer, 2007.
[10] N. Jazdi, C. Maga, P. Göhner, T. Ehben, T. Tetzner, and U. Löwen, "Mehr Systematik für den Anlagenbau und das industrielle Lösungsgeschäft - gesteigerte Effizienz durch Domain Engineering," *Automatisierungstechnik*, vol. 58, pp. 524–532, 2010.
[11] VDI/VDE, *VDI/VDE 3695: Engineering of Industrial Plants*, Beuth Verlag, 2013.
[12] S. Biffl, L. Arndt, and D. Gerhard, Eds., *Multi-Disciplinary Engineering for Cyber-Physical Production Systems, Data Models and Software Solutions for Handling Complex Engineering Projects*, Cham, Springer, 2017.
[13] VDI/VDE, *VDI/VDE 3682: Formalised Process Descriptions*, Beuth Verlag, 2005.
[14] J. Pfrommer, D. Stogl, K. Aleksandrov, S. Escaida Navarro, B. Hein, and J. Beyerer, "Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems," *Automatisierungstechnik*, vol. 63, pp. 790–800, 2015.
[15] N. Keddis, G. Kainz, and A. Zoitl, "Capability-based planning and scheduling for adaptable manufacturing systems," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation, ETFA 2014*, Barcelona, Spain, IEEE, 2014, pp. 1–8.
[16] S. Malakuti, J. Bock, M. Weser, et al., "Challenges in skill-based engineering of industrial automation systems*," in *23rd IEEE Int. Conf. On Emerging Technologies and Factory Automation, ETFA 2018*, Torino, Italy, IEEE, 2018, pp. 67–74.
[17] M. Weser, J. Bock, S. Schmitt, A. Perzylo, and K. Evers, "An ontology-based metamodel for capability descriptions," in *25th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA 2020*, Vienna, Austria, IEEE, 2020, pp. 1679–1686.
[18] P. Zimmermann, E. Axmann, B. Brandenbourger, K. Dorofeev, A. Mankowski, and P. Zanini, "Skill-based engineering and control on field-device-level with OPC UA," in *24th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA, 2019*, Zaragoza, Spain, IEEE, 2019, pp. 1101–1108.
[19] A. Köcher, A. Belyaev, J. Hermann, et al., "A reference model for common understanding of capabilities and skills in manufacturing," Automatisierungstechnik, 2023, In this issue.
[20] R. Capilla, J. Bosch, and K. C. Kang, Eds., *Systems and Software Variability Management - Concepts, Tools and Experiences*, Springer, 2013.

[21] K. Meixner, A. Lüder, J. Herzog, D. Winkler, and S. Biffl, "Patterns for reuse in production systems engineering," *Int. J. Software Eng. Knowl. Eng.*, vol. 31, pp. 1623–1659, 2021.

[22] K. Meixner, F. Rinker, H. Marcher, J. Decker, and S. Biffl, "A Domain-specific language for product-process-resource modeling," in *26th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA 2021*, Vasteras, Sweden, IEEE, 2021, pp. 1–8.

[23] F. Himmler, "Function based engineering with AutomationML - towards better standardization and seamless process integration inplant engineering," in *12th Int. Conf. on Wirtschaftsinformatik*, AIS Electronic Library, 2015, pp. 1216–1230.

[24] E. Maxwell, "Open standards, open source, and open innovation: harnessing the benefits of openness," in *Innovations: Technology, Governance, Globalization*, vol. 1, Cambridge, MA, MIT Press, 2006, pp. 119–176.

[25] R. Förster and F. Anna, "Einteilung der Fertigungsverfahren nach DIN 8580," in *Einführung in die Fertigungstechnik: Lehrbuch für Studenten ohne Vorpraktikum*, Berlin, Heidelberg, Springer, 2018, pp. 23–136.

[26] K. Meixner, A. Lüder, J. Herzog, H. Röpke, and S. Biffl, "Modeling expert knowledge for optimal CPPS resource selection for a product portfolio," in *25th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA 2020*, Vienna, Austria, IEEE, 2020, pp. 1687–1694.

[27] F. Rinker, K. Meixner, L. Waltersdorfer, D. Winkler, A. Lüder, and S. Biffl, "Towards efficient generation of a multi-domain engineering graph with common concepts," in *26th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA 2021*, Vasteras, Sweden, IEEE, 2021, pp. 1–4.

[28] E. Brandt, F. Brandt, K. Clemens, and D. Reichelt, "AI-supported marketplace for industrial capabilities," in *2021 22nd IEEE Int. Conf. on Industrial Technology, ICIT 2021*, Valencia, Spain, IEEE, 2021, pp. 1397–1402.

## Bionotes

**Dipl.-Ing. Kristof Meixner**
Christian Doppler Laboratory SQI,
TU Wien, Vienna, Austria
**kristof.meixner@tuwien.ac.at**

Kristof Meixner is a researcher at the Christian Doppler Laboratory SQI at TU Wien. His research interests include aspects of reuse and variability in capability- and skill-based production systems engineering. Kristof is a co-organizer of the yearly special session on capabilities and skills at ETFA.

**Dipl.-Ing. Felix Rinker**
Christian Doppler Laboratory SQI,
TU Wien, Vienna, Austria
**felix.rinker@tuwien.ac.at**

Felix Rinker is a researcher at the Christian Doppler Laboratory SQI at TU Wien. His research interests include aspects of multi-view engineering knowledge and change workflow management for Cyber-Physical Production Systems Engineering.

**Dipl.-Ing. Laura Waltersdorfer**
Semantic Systems Research Lab,
TU Wien, Vienna, Austria,
**laura.waltersdorfer@tuwien.ac.at**

Laura Waltersdorfer is a researcher investigating Auditable Semantic AI Systems in the medical science and environmental context. She has gained research experience in information systems and data integration approaches in industrial informatics, including quality assurance and process analysis.

**Prof. Dr. Arndt Lüder**
Institute of Ergonomics,
Manufacturing Systems and
Automation, Otto-von-Guericke
University, Magdeburg, Germany,
**arndt.lueder@ovgu.de**

Arndt Lüder is a professor in the field of factory automation leading the Institute of Ergonomics, Manufacturing Systems and Automation at Otto-von-Guericke University. His research focuses on the application of innovative technologies in factory automation covering control architectures and engineering methodologies.

**Prof. Dr. Stefan Biffl**
Institute of Information Systems
Engineering, TU Wien, Vienna,
Austria
**stefan.biffl@tuwien.ac.at**

Stefan Biffl is a professor of Software Engineering at TU Wien. His research interests include empirical software engineering, cyber-phycsical production system engineering, software process improvement, and software quality. He was leader of the 7-year research project Christian Doppler Laboratory FLEX.

## 5.3 Integrated Reuse and Variability Management for CPPS Engineering

### 5.3.1 Variability Transformation from Industrial Engineering Artifacts

#### Citation

[43] K. Feichtinger, **K. Meixner**, R. Rabiser, and S. Biffl. Variability transformation from industrial engineering artifacts: An example in the cyber-physical production systems domain. In R. Capilla, P. Collet, P. Gazzillo, J. Krueger, R. E. Lopez-Herrejon, S. Nadi, G. Perrouin, I. Reinhartz-Berger, J. Rubin, and I. Schaefer, editors, *SPLC '20: 24th ACM International Systems and Software Product Line Conference, Montreal, Quebec, Canada, October 19-23, 2020, Volume B*, volume Part F164402-B, pages 65–73. Association for Computing Machinery, 2020. doi: 10.1145/3382026.3425770

#### Aim

The publication "Variability Transformation from Industrial Engineering Artifacts: An Example in the Cyber-Physical Production Systems Domain" introduces the Variability Evolution Roundtrip Transformation (VERT) approach to transform from and to engineering artifacts with variability into state-of-the-art variability models using a transformation framework. The publication (i) presents a framework to transform CPPS engineering artifacts with variability to feature models and back, (ii) shows the feasibility of the framework using one of the previously presented real-world use cases.

#### Contribution to the thesis

This publication contributes to the research goals transformation framework for CPPS engineering artifacts with variability (**G3**., **G2**., **G4**.), and case study evaluation (**G5**.).

This publication contributes to **RQ1**. and **RQ3**. by addressing the VDI 3695 measures of *models and description languages* M1., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of interdisciplinary collaboration* C5..

#### Abstract

Many variability modeling approaches have been proposed to explicitly represent the commonalities and variability in (software) product lines. Unfortunately, practitioners in industry still develop custom solutions to manage variability of various artifacts, like requirements documents or design spreadsheets. These custom-developed variability representations often miss important variability information, e.g., information required to assemble production goods. In this paper, we introduce the VERT process. The process enables practitioners from the CPPSs domain to transform custom-developed engineering

variability artifacts to a feature model, evolve and optimize the model, and transform it back to the original engineering artifacts. We build on an existing transformation approach for variability models and show the feasibility of the process using a real-world use case from an industry partner. We report on an initial feasibility study conducted with our industry partners' domain experts and on lessons learned regarding variability transformation of engineering variability artifacts.

# Variability Transformation from Industrial Engineering Artifacts: An Example in the Cyber-Physical Production Systems Domain

Kevin Feichtinger
LIT CPS Lab
Johannes Kepler University Linz, Austria
kevin.feichtinger@jku.at

Kristof Meixner
Christian Doppler Lab CDL-SQI, ISE
TU Wien, Austria
kristof.meixner@tuwien.ac.at

Rick Rabiser
LIT CPS Lab
Johannes Kepler University Linz, Austria
rick.rabiser@jku.at

Stefan Biffl
Inst. of Information Systems Eng.
TU Wien, Austria
stefan.biffl@tuwien.ac.at

## ABSTRACT

Many variability modeling approaches have been proposed to explicitly represent the commonalities and variability in (software) product lines. Unfortunately, practitioners in industry still develop custom solutions to manage variability of various artifacts, like requirements documents or design spreadsheets. These custom-developed variability representations often miss important variability information, e.g., information required to assemble production goods. In this paper, we introduce the *Variability Evolution Roundtrip Transformation* (VERT) process. The process enables practitioners from the *Cyber-Physical Production Systems* domain to transform custom-developed engineering variability artifacts to a feature model, evolve and optimize the model, and transform it back to the original engineering artifacts. We build on an existing transformation approach for variability models and show the feasibility of the process using a real-world use case from an industry partner. We report on an initial feasibility study conducted with our industry partners' domain experts and on lessons learned regarding variability transformation of engineering variability artifacts.

## CCS CONCEPTS

• **Software and its engineering** → **Software product lines**.

## KEYWORDS

Variability Modeling, Feature Extraction, Variability Evolution, Cyber-Physical Production System Engineering, CPPS

**ACM Reference Format:**
Kevin Feichtinger, Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2020. Variability Transformation from Industrial Engineering Artifacts: An Example in the Cyber-Physical Production Systems Domain. In *24th ACM International Systems and Software Product Line Conference Companion (SPLC '20 Companion), October 19–23, 2020, MONTREAL, QC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3382026.3425770

## 1 INTRODUCTION

Variability modeling is the discipline of explicitly representing variability in dedicated models that describe the common and variable aspects of a set of (software) systems [9] as the basis for Software Product Line (SPL) engineering [32]. Several works have focused on comparing existing variability modeling approaches, tools, and languages [4, 7, 12, 19, 33, 39] and on analyzing their use in industry [9]. Still, there is currently no unified or standardized variability modeling approach available. Industrial practitioners are often not familiar with academic approaches from the SPL community [34] and develop custom solutions to manage variability instead of adapting existing approaches [9, 19]. While existing approaches may have their benefits and limitations, they still could be a useful basis to manage variability in their particular domain and use case.

For this paper, we worked with a leading company in the integration of high-speed automation for Cyber-Physical Production Systems (CPPSs). The industry partner's domain experts represent variability information on CPPS requirements in different artifacts, e.g., *type comparison matrices (TCMs)* (product parts vs. product types) and CAD drawings of product types. Unfortunately, these variability representations are incomplete and require additional information, like production constraints, to derive production processes and corresponding CPPS design correctly. This information is individual implicit knowledge of domain experts. The lack of a systematic variability modeling approach and implicit variability knowledge make the evolution of product lines time-consuming and error-prone. In workshops with CPPS domain experts, we identified the following two research challenges.

**Challenge 1.** *No process available for evolving engineering variability artifacts while using the benefits of well-known variability models.* In CPPS engineering, *basic engineers* draft initial CPPS designs in the *basic planing* phase, i.e., for a rough cost estimation, before handing them over to *detailed planning*. Therefore, they need to understand the impact of product requirements on the sequence of production processes and the particular CPPS designs. Important product requirements come from additional product types, added to the product line over time, i.e., an evolving product line.

The variability information within the custom solutions is typically spread across various artifacts, like requirements documents or spreadsheets, and across different domain experts. This information dispersion makes understanding the impact of changes and evolution difficult. Hence, a process to evolve industrial variability artifacts while using the benefits of a well-known variability modeling approach, e.g., feature modeling, would help basic engineers to evolve their variability representations and, consequently, understand the impact of their designs better.

Unfortunately, such a process is missing as domain experts (a) mostly apply implicit knowledge and heuristics and do not know how to (b) elicit the relevant variability information from the engineering artifacts and (c) transform this variability information back to the tools they require, e.g., for cost estimation.

**Challenge 2.** *Unclear how to implement a transformation of custom-developed industrial engineering variability artifacts into well-known variability models.* The variability information in engineering artifacts is often held in differing granularity, making it challenging to systematically analyze and model variability. So even when a well-known variability model could be used, it is hard for domain experts to model their variability in the new approach. It often fails due to the implicit knowledge about existing variability representation and the ambiguity of extracting features correctly from existing representations.

Our industry partner's basic engineers have problems extracting features from their variability representations in TCMs and CAD drawings and capturing their implicit knowledge about the product variability in a well-known variability model. Therefore, a structured method to transform these custom-developed variability representations into a variability model, e.g., a feature model, would support basic engineers to model their variability and further make implicit knowledge more explicit. However, we do not expect the domain experts in CPPS engineering disciplines to switch to a variability modeling approach soon entirely. Transforming between different variability representations to employ their particular benefits would help them understand variability better and support them in the evolution of their representations.

From the challenges, we raise two research questions:

**RQ1.** *How can a process support CPPS engineers evolving a product line represented in industrial CPPS engineering variability artifacts while taking advantage of well-known variability models?* To address Research Challenge 1, we introduce and validate the Variability Evolution Roundtrip Transformation (VERT) process (cf. Figure 1 and Section 4) to (a) transform industrial variability artifacts into a feature model, (b) refine the resulting feature model based on domain expert feedback making implicit variability knowledge explicit, and (c) evolve the feature model and industrial variability artifacts according to additional CPPS requirements. We conducted the VERT process in an initial feasibility study. We discussed the resulting feature model and artifacts with our industry partner's domain experts to evaluate the process and artifacts' utility and efficacy. In this paper, we report on the results of the evaluation and lessons learned during the process.

**RQ2.** *How can custom-developed engineering variability artifacts be (semi-)automatically transformed into a well-known variability model?* To address Research Challenge 2, we adapted a transformation approach for variability models [18] to automatically transform custom-developed industrial variability representations into an initial feature model as a basis to incorporate domain expert feedback and then transform it back to the original variability representations. This paper introduces transformation operations implemented in transformation algorithms, which we use in the VERT process, as well as lessons learned.

The remainder of this paper is structured as follows. Section 2 discusses background and related work on CPPS, transformation of variability models, and feature extraction. Section 3 introduces the industrial *shift fork* product line as the use case for the initial feasibility study. Section 4 presents the VERT process steps and artifacts as well as transformation operations for engineering variability artifacts of our industry partner. Section 5 discusses the initial feasibility study with feedback from the industry partner and lessons learned. Section 6 concludes the paper and outlines future work directions.

## 2 BACKGROUND AND RELATED WORK

This section summarizes work on CPPS engineering, transforming variability models, and feature extraction.

### 2.1 Cyber-Physical Production Systems Eng.

CPPSs are envisioned as next-generation production systems aiming for flexible and optimized production of customized goods. CPPSs employ autonomous and cooperative resources capable of interacting with their environment and utilize modern manufacturing techniques combined with information and communication technology [28]. Key characteristics of a CPPS are, e.g., robustness, real-time control, and self-adaptive behavior to uncertain conditions [21, 28]. Examples range from automated car manufacturing plants to large-scale smart grids, which can re-allocate resources under load.

Engineers from different domains design CPPSs in consecutive engineering phases, creating a multidisciplinary environment [10]. A proficient collaboration of engineers in such environments is key but requires a sufficient exchange of engineering knowledge between domains and throughout the CPPS lifecycle. Model-based, machine-readable, and easy to exchange engineering knowledge representations facilitate such a knowledge exchange [8].

The three main aspects of CPPSs are *products*, manufactures by *processes* that employ *resources* to transform the state of the product parts [38]. Realizing the vision of flexible production of product lines requires the management of variability of the three aspects at engineering and run time. The variability in CPPSs stems from (a) *product types* and their features, (b) *process sequence variants* and their particular characteristics, and (c) *resource candidate variants* that enable executing a process task. To that effect, the variability of the Product-Process-Resource (PPR) aspects and their dependencies create a large and complex problem and configuration space. Depending on the architecture of the CPPS, e.g., assembly line and/or work cell production, engineers have to make design decisions that define the CPPS layout and production sequence at particular phases, which can be understood as the binding time for CPPS.

In this work, we focus on design decisions implied by product types' structure and their assembling order in a CPPS.

### 2.2 Transformation of Variability Models

Many different approaches have been developed for managing variability. The most common approaches are feature modeling and decision modeling [14]. Researchers have compared variability modeling approaches [7, 14, 17] and investigated transforming between different approaches, e.g., feature model to Orthogonal Variability Modeling (OVM) [35], or transforming feature models into propositional formulas or binary decision graphs for automated analyses [5, 15]. Although there were and still are ongoing efforts to develop a standard variability modeling language, e.g., CVL [20], currently, no such language exists.

This work builds on a transformation approach for variability models [18], which allows the transformation of an industrial variability representation into a well-known variability model and vice versa. In our case, we transform the industrial variability representations, consisting of a set of artifacts, into a feature model. The approach consists of three main components. *Variability meta-models* capture the (types of) elements and dependencies of variability models. Each particular type of variability model has a meta-model, which either exists explicitly [11] or implicitly [22], and/or an implementation in a tool [16]. *Transformation operations* rely on the meta-meta models of the different variability model types and describe the required operations to transform one (type of) variability model into a different one (e.g., feature model to decision model). *Transformation algorithms* rely on the transformation operations and the meta-models of two variability model types and implement how one concrete type of variability model is transformed into another concrete type of variability model (e.g., FeatureIDE model [25] to DOPLER model [16]).

For this paper, we derive transformation operations and implement them in transformation algorithms to derive a feature model from our industry partner's custom-developed variability representation and vice versa. These transformations allow an integration to our industry partner's existing processes while still benefiting from the feature modeling capabilities. It also gives us the flexibility to work with different artifact types and generate different model types in the future.

### 2.3 Feature Extraction

Retrieving the variability of existing systems can be a tedious task, especially when done manually. Several *Feature Extraction and Identification* approaches have been proposed to automatically reveal and analyze variability in various artifacts [3, 13, 24, 27].

For product comparison matrices (PCMs), which are quite similar to TCMs, some approaches exist. Nasr et al. [29] developed an approach to extract PCMs from pre-selected product descriptions in natural language and provided evidence that the synthesized PCMs sufficiently represent product variability. Bécan et al. [6] propose a meta-model for a more formal PCMs representation and an automated approach to transform PCMs to this model. In [36, 37], the authors identified that unclear semantics are one of the major limitations of PCMs. The authors propose using variability models over PCMs, emphasize the drawbacks of current PCMs feature extraction methods, and raise questions to bride the gap between PCM and variability models. In a mapping study, Assunção et al. [2] discuss further works on re-engineering of systems into product lines with a focus on transforming various artifacts into reusable components.

Most approaches typically focus on specific types of artifacts and can only generate one kind of variability model. This specialization makes it hard to apply the approaches in an industrial context, across a heterogeneous set of variability representations used in different disciplines, potentially evolving in the future.

## 3 THE SHIFT FORK USE CASE

Our industry partner is a leading company in the integration of high-speed automation for CPPSs. The company plans and engineers CPPSs based on the requirements of customers that want to manufacture product lines, like automotive parts, effectively and efficiently. In the following, we outline the activities and artifacts that engineers execute and create in the basic planning phase, resulting in a rough layout of the CPPS and a corresponding cost estimate, using the *shift fork product line* use case. A shift fork is part of a manual transmission in a machine, e.g., a car, that shifts a cuff[1] along rods into a particular position so that the gears connect correctly. Many parts of the shift fork types are similar but, depending on the type, need to be installed at different positions.

At the beginning of the CPPS engineering process, *basic engineers* receive designs and/or prototypes of different product types from the customer. The engineers investigate different options for assembling the particular product types. Hence, they explore the product variability and dependencies between product types, their parts, and assembly groups[2]. They then enter the identified *is part of* relations into a *type comparison matrix (TCM)* (cf. product comparison matrix [29]) to document the requirements and calculate, e.g., the number of required parts for a particular number of final products. Table 1 shows a TCM for the shift fork use case with four shift fork types and 14 parts. The TCM holds product types in the columns and parts and assembly groups in the rows.

Basic engineers use implicit company-specific semantics to create the TCM. In this use case, an $x$ in the type column and part row marks that a product type requires the particular product part. For instance, *Fork-13* requires *Barrel 1*, a commonality of all product types, pipe *Pipe 8*, and lock *Lock 3*, which differ among the product types. However, across engineering projects, TCMs may slightly differ in their semantics. For instance, if a product contains two instances of the same type, e.g., *Barrel 1*, an engineer may write 2 in the row of a part type or create two rows, each containing the same part type. Secondly, a TCM can get quite large with typically consisting of 45 columns and 300 rows. The inconsistent semantics and the often huge table makes it even for engineers challenging to read this engineering artifact. From the TCM and implicit knowledge on assembly techniques, the basic engineers identify the abstract production processes and resources of the CPPS. Then, they design a first draft of the production process sequence for the unification of all processes for the requested product types with a corresponding CPPS layout and a cost estimate.

From discussions with domain experts at our industry partner, we assume that a more formal model with defined semantics, like a feature model, would benefit their work. It could help them to analyze and discuss the features and their dependencies. Furthermore, an explicit representation of the product parts' dependencies and how they are assembled would improve the knowledge transfer

[1]See a shift fork illustration at https://commons.wikimedia.org/wiki/File:Manual_transmission_clutch_First_gear.PNG
[2]An assembly group is an intermediate part that can be assembled in a separate CPPS module or in parallel to other tasks.

**Table 1: Variability matrix of the *shift fork* types.**

| Parts/Types | Fork-13 | Fork-2R | Fork-46 | Fork-57 |
|---|---|---|---|---|
| Pipe 2 | | | | × |
| Pipe 3 | | × | × | |
| Pipe 8 | × | | | |
| Barrel 1 | × | × | × | × |
| Jack 1 | × | × | × | × |
| Ring 1 | × | × | × | × |
| Fork 3 | × | × | × | × |
| Fork 4 | × | × | × | × |
| Fork 5 | × | × | × | × |
| Lock 1 | | | × | × |
| Lock 2 | | × | | |
| Lock 3 | × | | | |
| Screw | × | × | × | × |
| O-Ring | × | × | × | × |

between engineering phases and disciplines, especially as some of the domain experts' knowledge is currently only available implicitly, as internal expertise. Still, experience [9, 41] shows the difficulty to motivate a company to switch to a new approach (and tooling), in part due to the effort to create and maintain variability models manually. We thus aim to (1) automatically generate at least an initial version of a feature model and (2) to support *roundtrip engineering*, i.e., allow domain experts to keep working with their artifacts while updating them based on feature model changes, like refined constraints.

## 4 VARIABILITY TRANSFORMATION FROM AND TO ENGINEERING ARTIFACTS

This section introduces the *Variability Evolution Roundtrip Transformation (VERT)* process and the transformation operations to transform the engineering variability artifacts into a feature model and vice versa.

### 4.1 Variab. Evol. Roundtrip Transform. Process

We define the six-step VERT process (cf. Figure 1) to address research question *RQ1*. The process is inspired by agile engineering techniques and developed based on discussions with several industry partners from CPPS engineering as well as recent research on variability model transformation [18]. The VERT process starts from engineering artifacts provided by the basic engineers and transforms these custom-developed variability representations into an initial feature model. This feature model serves as the basis for an optimization and evolution cycle using configuration sampling and basic engineers' feedback. The optimized feature model is then transformed back into their original variability representation. This roundtrip allows basic engineers the evolution of the product and CPPS variability more systematically and make the implicit knowledge available. In the following, we describe the VERT process steps in detail and illustrate them using the *shift fork* use case.

**Step 1:** *Analyze variability information.* In this step, the *variability modeling expert (VME)* (in our case, a researcher) analyzes various engineering artifacts to understand the product types, their variability, and dependencies. These artifacts, provided by the basic engineers, consist of initial TCMs, engineering documents, e.g., CAD drawings, and basic engineering knowledge. The analysis of the artifacts results in a cleaned-up *TCM* (cf. Table 1) and a *Precedence Graph (PG)* (cf. Figure 2) with the relevant variability information extracted. The initial TCMs often contain additional information, like pictures, word glossaries, and various comments. Hence, the VME needs to clean up these initial TCMs and extract the relevant variability information. The result is a TCM representing the different product parts and the product types, similar to the one in Table 1. The TCMs, as provided, contain only limited information on dependencies between the particular product parts of a product type. The remaining information is implicit knowledge of basic engineers, for example, *implies* relations and a basic sequence of assembly steps (e.g., that the *fork* needs to be welded onto the *pipe*). The PG represents these dependencies explicitly as a basis for reasoning on possible product structures and assembly sequences.

Figure 2 illustrates for the shift fork use case a PG that defines which parts need to be installed before particular other parts can be installed[3] as black-headed arrows. For instance, *Fork 5* needs to be welded onto the *Pipe* prior *Fork 3* and *Fork 4* as, otherwise, the welding point cannot be reached anymore. However, it is arbitrary whether *Fork 5* or *Barrel 1* is installed first. The graph also shows which product parts are subtypes of an abstract group, in gray, using white-headed arrows. For example, *Pipe 2* belongs to the *Pipe* group as *Lock 1* belongs to the *Lock* group. The VME creates an $N \times N$ adjacency matrix that represents the PG as shown in Figure 2 from the given variability descriptions, where $N$ is the number of parts in the product line.

**Step 2:** *Transform variability artifacts to feature model.* In this step, the VME uses our transformation approach to automatically transform the TCM and PG from Step 1 into an over-approximated feature model (cf. Figure 3). The generated feature model allows to conduct advanced analyses of the product types and their dependencies. Subsection 4.2 describes the transformation operations extracting features from the TCM and the PG to create an over-approximated feature model in detail.

**Step 3:** *Derive configuration samples.* In this step, the VME uses a configuration generator that follows a sampling algorithm to derive configuration samples, i.e., product type samples. Configuration sampling is a well-known technique where a representative set of valid configurations [30, 31] is created to enable product-based testing of software product lines [40]. In our process, we rely on the YASA [23] sampling algorithm implemented in the FeatureIDE [25].

The generated product type samples from this step are based on the over-approximated feature model coming from Step 2 or the improved feature model coming from Step 5. The product type samples are a valid set regarding the feature model and allow checking whether it fits the required set of product types.

**Step 4:** *Collect feedback from basic engineer.* In this step, the VME and basic engineers, based on their implicit knowledge, manually analyze and discuss the initial feature model in comparison with the product type samples to collect feedback in a shared document for improving the model. The VME and basic engineers take the over-approximated feature model and the product type samples coming from Step 3 as input for analysis. The basic engineers give

[3]For better visualization, we do not show the transitive dependencies in the figure.

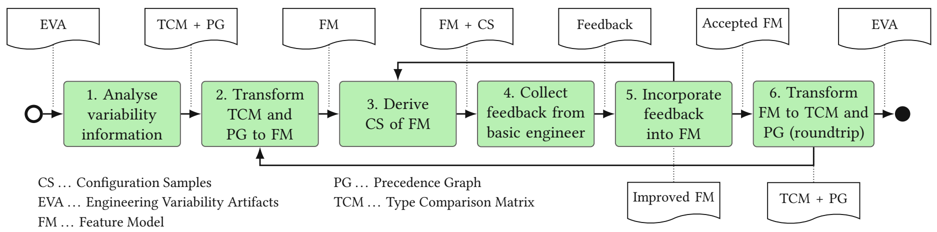

EVA
TCM + PG
FM
FM + CS
Feedback
Accepted FM
EVA
1. Analyse variability information
2. Transform TCM and PG to FM
3. Derive CS of FM
4. Collect feedback from basic engineer
5. Incorporate feedback into FM
6. Transform FM to TCM and PG (roundtrip)
Improved FM
TCM + PG
CS … Configuration Samples
EVA … Engineering Variability Artifacts
FM … Feature Model
PG … Precedence Graph
TCM … Type Comparison Matrix

**Figure 1: The *VERT* process of generating a feature model from custom-developed variability representations.**

feedback on the overall feature model structure, e.g., number and correctness of features and feature groups. The basic engineers then check whether the product type samples fit the customers' product line and if the most relevant product types exist. They also check if the constraints in the feature model are too strict or too loose. The feedback loop results in a proposal of changes to the feature model, e.g., to add or rearrange features and improve constraints, or the acceptance of the feature model. The feedback also pertains to improvements of the engineering variability artifacts, such as inconsistencies or missing information. For evaluating the VERT process, we conducted this step with three of our industry partners' domain experts for the shift fork use case (cf. Section 5).

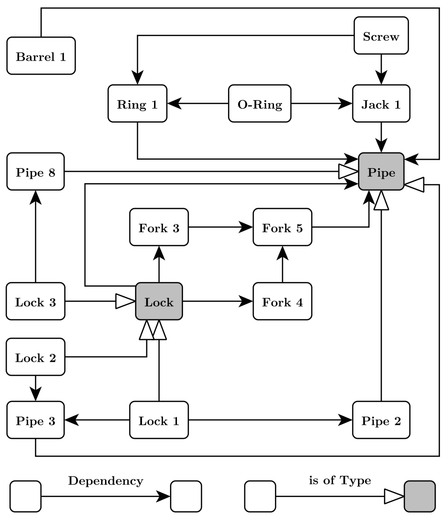


**Figure 2: The *Precedence Graph* (PG) of *shift fork product line* parts from the TCM (cf. Table 1).**

**Step 5:** *Incorporate feedback into feature model.* In this step, the VME incorporates the feedback of the basic engineers from Step 4 into the feature model. Therefore, either the VME integrates the required changes into the feature model resulting in an improved feature model, or the basic engineers approve the model resulting in an accepted feature model. The improved feature model is input to the next refinement iteration in Step 3, where product type samples are generated based on the model. The accepted feature model is forwarded to Step 6. Figure 4 shows the accepted feature model with several modifications requested by the domain experts. The feedback cycle between Step 3 and Step 5 can be executed directly in the meeting with the basic engineers with software support like feature modeling tools.

**Step 6:** *Generate engineering variability artifacts (roundtrip transformation).* In this step, the VME uses the transformation approach to generate the evolved engineering variability artifacts from the accepted feature model. The results are a new TCM and PG with the changes of the feature model. This way, basic engineers can effectively pass on the feedback collected (Step 4) and changes made in the feature model (Step 5) to the original engineering variability artifacts. Furthermore, basic engineers can work with the generated engineering variability artifacts, e.g., to calculate certain product properties or evaluate a CPPS design. The adapted engineering variability artifacts can then be used as an input to Step 2 for another iteration closing the roundtrip and as a basis for the co-evolution of domain-specific variability artifacts and feature models.

## 4.2 Variability Transformation Operations

We define a sequence of transformation operations that allow the transformation of the TCM and PG into a feature model and vice versa, to address research question *RQ2*. The transformation operations aim for an over-approximated feature model, as redundant information can be removed during the first feedback session with the domain expert, but might help them to understand the resulting model. We implemented the transformation operations to automatically transform the TCM and PG into a FeatureIDE feature model [25] and back. The algorithms take the TCM and PG as input and produce a feature model, or take the feature model as input and create a TCM and PG, respectively. Figure 3 shows the generated feature model from the custom variability representation. We conducted a preliminary evaluation with data from the

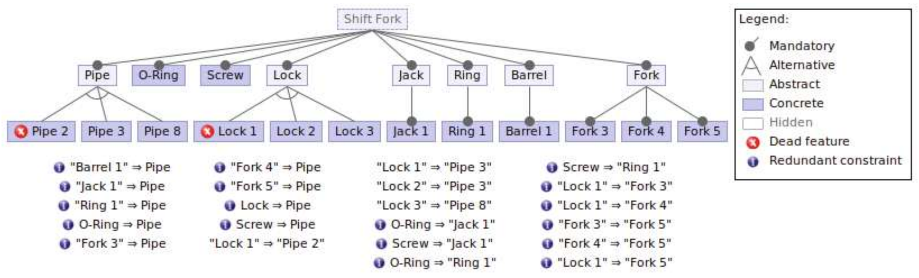


**Figure 3: The initially generated feature model for the *shift fork product line* consists of 21 features and 22 constraints with a maximal feature tree height of 2.**

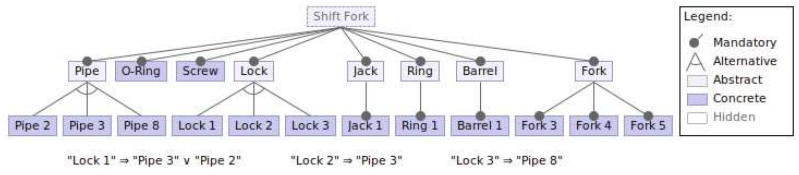


**Figure 4: The updated feature model for the *shift fork product line*, accepted by domain experts after four iterations. The accepted feature model consists of 21 features and 3 constraints with a maximal feature tree height of 2.**

shift fork use case (cf. Section 5) to evaluate the feasibility of these transformations.

*Transformation from TCM and PG to feature model.* Operation **productPartToFeature** creates a feature for each part in the TCM. For a group of parts sharing the same name (e.g., feature *Pipe 2*, *Pipe 3* or *Pipe 8* in Figure 3) the operation creates an abstract parent feature (cf. feature *Pipe* in Figure 3). This approach works in practice, as similar product parts, like the pipes, have names with part numbers that start with the same number groups. Operation **productTypeToFeatureCardinality** defines the cardinality of each feature according to the types it is part of. Per default, each feature is *optional* but becomes *mandatory* if it is part of all product types. If there is only one instance of a part per product type, the feature group becomes an *alternative* group (e.g., only one *Pipe* is used at the same time). If a feature group is not used in all but used together with other parts of the same group, the feature group becomes an *or* group (not visible in the example use case). Operation **productTypeToFeatureAttribute** defines that each part-feature stores its relevant product types as attributes of the feature (e.g., store *Fork-2R* and *Fork-46* as attributes in feature *Pipe 3*). Type information stored with the derived features enables *roundtrip engineering artifact* transformation. Operation **dependencyToImpliesConstraint** transforms the dependencies captured in the PG into *implies* constraints in the feature model. For each dependency, it creates an *implies* constraint, based on the direction of the dependency in the PG. For instance, the *implies* constraint Lock 1 implies Pipe 3 is based on the dependency Lock 1 → Pipe 3 in the PG (cf. Figure 2). Many constraints derived from the PG would not be necessary, as those constraints are already represented via feature groups in the feature model. Nevertheless, these constraints may help domain experts understand the model and are thus kept. They can be removed in a feedback meeting or using optimization support from FeatureIDE. The over-approximated constraints in the feature model impact the set of product configurations that can be derived from the feature model, as some of the constraints violate the feature group constraints (e.g., constraints Lock 1 → Pipe 2 and Lock 1 → Pipe 3 violate the alternative constraint of the Pipe feature group). Therefore, feedback from basic engineers is necessary to capture the product variability correctly.

*Transformation from feature model to TCM and PG.* Operation **featureToProductPart** transforms each part-feature of a feature model into a product part of the TCM. Operation **featureAttributeToProductType** restores the product types of the TCM from the feature attributes. Specifically, it restores for each part the types from the respective feature and stores these type assignment in the TCM. Operation **constraintToPGDependency** restores the PG from the constraints of the feature model. The operation creates a dependency for each *implies* constraint in the resulting PG.

## 5 PRELIMINARY EVALUATION

We conducted an initial feasibility study on the *Variability Evolution Roundtrip Transformation (VERT)* process and the implemented transformation operations as a preliminary evaluation and to collect industry feedback on the research results.

The study's objective is to assess the utility and efficacy of the VERT process and the artifact transformation from a domain expert's perspective. Two researchers (authors of this paper) conducted the VERT process in a workshop together with a basic engineer and two senior domain experts at our industry partner to investigate how they handle the process and value the artifacts. They employed the *shift fork* use case and conducted the process in four iterations. We assume that the VERT process helps domain experts to better model and evolve their variability representations and reveals implicit knowledge that can be externalized. Finally, two researchers (authors of this paper) collected the senior domain experts' feedback during the workshop and discussed the interviews' results after finishing the process. They mainly collected feedback on the representations of the transformation results and input for future work. In the evaluation, we focus on the research questions (RQs) raised in Section 1. For both questions, we distilled industry feedback takeaways (Fx.x – words in brackets serve for better understanding) and lessons learned (LLx.x) for researchers.

### 5.1 Var. Evo. Roundtrip Trans. Process (RQ1)

The domain experts found the VERT process understandable and useful to augment their current CPPS engineering process. The domain experts found the VERT variability model representations, i.e., the TCM and the PG useful. They noted that these artifacts capture the dependencies of the custom-developed variability representation and suitably express implicit domain expert knowledge to facilitate analysis and reuse. In particular, knowledge reuse concerns understanding the impact of changes to the product line that require the evolution of variability knowledge. We received the following feedback for future work.

**F1.1:** *Co-evolution of variability representation and product type changes is a key capability.* A major takeaway was the importance of a structured and systematic co-evolution of product type and CPPS variability representation. In practice, evolutionary changes are triggered by new product types or processes or by changing product types and processes. Basic engineers "often do not dare to change designs made by other domain experts but rather design many artifacts from scratch to avoid the risk of corrupting a cost estimate or running system". The engineers noted that a method to use a "feature model [to represent their variability needs] is definitely worth the additional effort for creating the [variability] model." This emphasizes the need for further research in the field of variability co-evolution in CPPS engineering.

**F1.2:** *The variability model has to represent production processes.* The basic engineers pointed out that "at some point, we also require the [assembly] processes [that produce the products] to be shown in the feature [variability] model" (cf. Section 2.1) to represent CPPS variability sufficiently. For instance, a particular process that does not work for a product type as initially intended, e.g., a screw cannot be easily installed from a designed position, may require changing the product assembly groups and the feature groups and constraints in the feature model. This coexistence of engineering artifacts requires further research on a variability model that integrates product, process, and resource variability [26].

Besides the feedback of the domain experts, we learned the following lessons during the research process.

**LL1.1:** *Understanding custom variability representations is challenging.* One of the biggest challenges for researchers was understanding the custom variability representations (such as TCMs, CAD drawings of products, and CPPS layouts) in VERT Step 1 well enough to design the PG and systematic operations to transform these variability representations into a feature model. Without the knowledge of and training by domain experts, it would have been very difficult to create a feature model that domain experts can work with. Sometimes even the domain experts struggled to remember and express the reasons for particular design decisions.

**LL1.2:** *The level of detail in the Precedence Graph determines the semantic foundation for expressing constraints.* The variability model's goal is to allow configuring all valid product types that the CPPS shall be able to produce. We experienced that one must be careful *not to over-constrain the model* using the PG. In our case, the level of detail of the shift fork's PG and the derivation of the adjacency matrix had a major impact on the feature model's "configurability". Suppose there are too many details and dependencies in the matrix, e.g., not grouped around parent type. In that case, very few product configurations can eventually be derived from the resulting feature model, or the feature model may become invalid. Hence, besides the careful creation of the PG, the definition of suitable rules to derive its adjacency matrix needs to be investigated before extracting the constraints. This issue is based on the fact that the engineers currently do not create such an explicit artifact. We also experienced that the adjacency matrix *as-is* cannot express excludes or negations constraints. This calls for an extension of the semantics of the PG and adjacency matrix.

Our preliminary evaluation results show the feasibility and indicate the utility of our VERT process as a foundation for product line evolution (cf. RQ1) by systematically transforming industrial variability engineering artifacts to a well-known variability model.

### 5.2 Variability Transformation Alg. (RQ2)

The domain experts liked the feature model generated by the variability transformation algorithm and the product variability representation as a feature model in general. The FeatureIDE [25] supported them in easily adding new product features and dependencies. Further, domain experts found it useful to transform the updated feature model back into the original representation of the TCM and the PG as input to their subsequent processes. They gave the following valuable feedback for future work on the transformation and the resulting feature model.

**F2.1:** *Counter-intuitive syntax of the feature model and constraints.* Domain experts had problems to understand the syntax of the feature model and its constraints at first. They especially experienced problems with particular constraints when incorporating the feedback to the feature model despite configurations sampling. For example, they were unsure whether "the lock should imply the pipe" or the other way around, making it hard for them to resolve dead features in the feature model. Teaching variability modeling (to industry) is a foundation for practical application the community needs to improve [1].

**F2.2:** *Integration of variability artifacts into one feature model is useful.* Figure 3 shows the feature model generated from the TCM (see Table 1) and the PG (see Figure 2). The domain experts liked "the hierarchical structure of the feature model and the concept of abstract features to group features to assembly groups" used in different product types. Integrating variability information from multiple artifacts into one single model was considered useful.

**F2.3:** *The feature model should contain product type information.* In comparison to the engineering artifacts, the feature model loses the information of the particular product types, which are crucial to track for the domain experts. CPPS engineers think in product types that they receive as requirements from customers. While variability models provide support to represent the commonalities and variability of the product types, only very few product configurations make sense to produce on a CPPS. The engineers want "at any point in the engineering process an overview on the product types that are modeled in the variability model". This is a big difference in software engineering. The "production cost" of combining features to software products is small compared to producing industrial goods, which has to focus on profitable types.

**LL2.1:** *Custom variability representations lack well-defined semantics.* For example, certain product parts are used multiple times during product assembly (e.g., *Barrel 1* is used twice in several shift forks). However, this multiplicity is not always reflected as proper quantification in the engineering artifacts. When asked why their variability representation does not capture this information, domain experts mainly gave two reasons. First, unfortunately, they do not have rules for treating instances in the variability representation, but "rely on heuristics and the experience of the experts on what works best for a particular project". Second, sometimes the customer provides a basic type comparison matrix along with other requirements documents and the basic engineer adopts the customer's informal semantics. This confirms research reported by Sannier et al. [36] and exemplifies the issue of insufficient semantics also in industry.

**LL2.2:** *Tool support is essential in feature extraction approaches.* The results of the VERT evaluation indicate a promising first step towards variability co-evolution in CPPS engineering. The automatically generated initial variability model is a good basis for further discussions. However, further specific roundtrip and co-evolution tool support is needed for practitioners. Our industry partner aims to provide basic engineers with novel engineering tools that shall build on capabilities from recent research on variability modeling and co-evolution support.

Our preliminary evaluation demonstrates that a set of transformation operations could be successfully implemented to (semi-) automatically transform industrial variability engineering artifacts into a variability model and back (cf. RQ2).

## 6 CONCLUSIONS AND FUTURE WORK

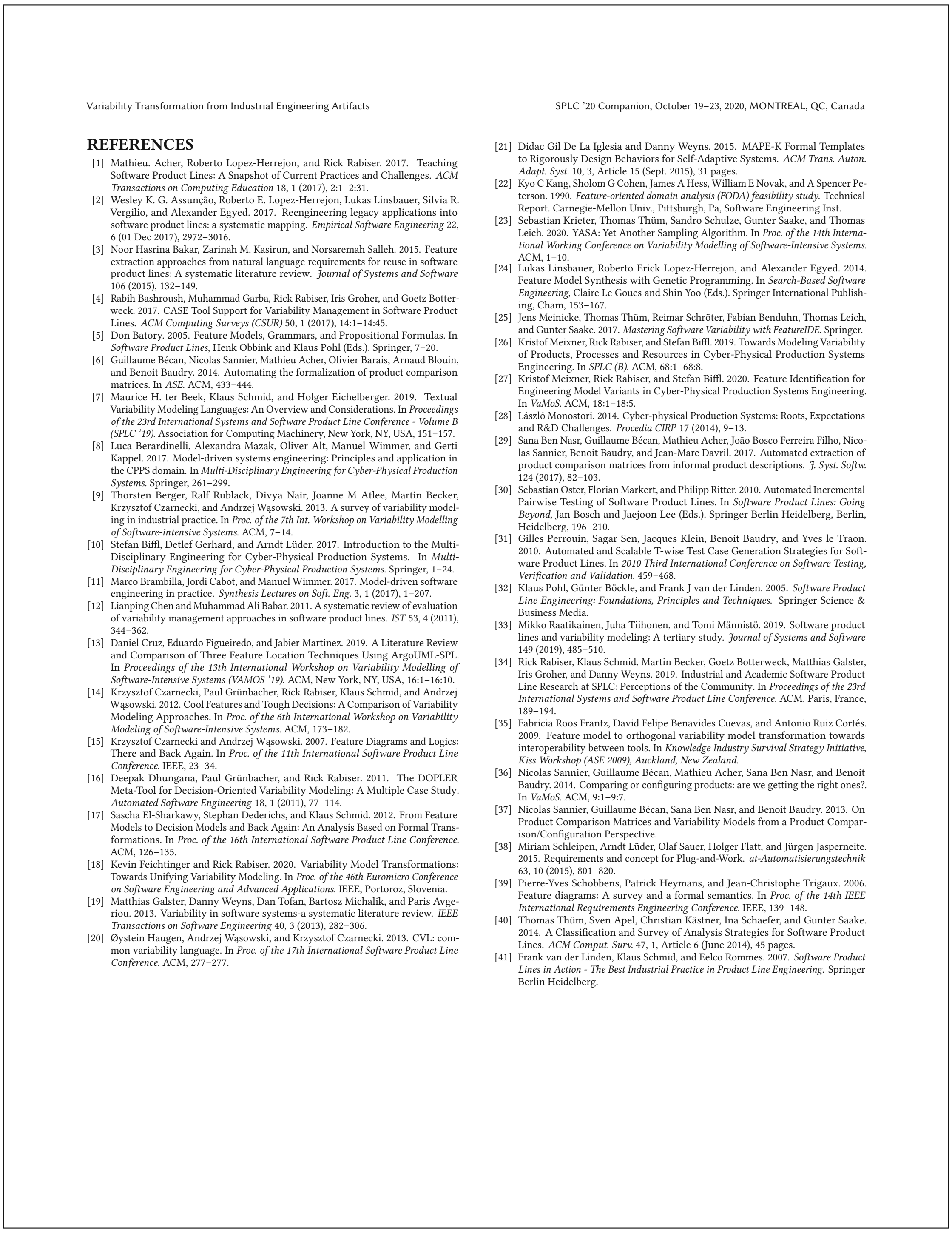
Practitioners frequently develop custom variability representations that hardly follow or adapt one of the many systematic variability modeling approaches from the software product line community. These custom representations are often incomplete, with weak semantics, e.g., spreadsheets, and lack domain experts' implicit knowledge on variability. In this paper, we introduced the iterative *Variability Evolution Roundtrip Transformation* process, enabling the systematic transformation and evolution of engineering variability artifacts that represent related product types produced on *Cyber-Physical Production System* to a feature model. We built on an existing transformation approach to transform the artifacts. We showed the VERT process' feasibility for the representative industrial *shift fork product line* use case. We evaluated the VERT process with our industry partner's domain partner on utility and efficacy and reported on their feedback and lessons learned for researchers.

Domain experts found the VERT process feasible and useful, and the generated feature model suitable in the study context. We infer that the VERT process is a promising contribution towards product variability evolution in CPPS engineering. We found that the impacts of product line evolution on CPPS process and layout design can be significant and that understanding them is crucial for domain experts. Hence, we advocate for an integrated CPPS variability model considering the variability of products, processes, and resources alike. Understanding the semantics of engineering variability artifacts and grasping implicit domain experts' knowledge is challenging. In reverse, the semantics of well-known variability models seem to not always be intuitive for domain experts. We argue that industry and academia do not understand their respective concepts well enough to find common ground, calling for closer collaboration to convey relevant concepts. Yet, transforming between variability models can bridge a gap between the fields. We also found that product type information is significantly more important in the engineering domain than the software domain, reflecting fundamentally different needs of the two domains.

In *future work*, we plan to investigate using type comparison matrices (TCMs) from larger projects and examine the evolutionary impact of omitting and adding product types to a product line one at a time. We further plan to use a Precedence Graph (PG) to represent the variability of (production) process sequences and derive and inform basic engineers on feasible process alternatives, e.g., production processes optimized for resource type usage. Such a PG could then be transformed into a *decision model* for design decision support and to generalize the employed transformation approach. Based on these experiences, we plan to perform a case study with an industry partner to evaluate the transformation approach. Additionally, an extension of the adjacency matrix semantics to represent implies, excludes, and negation constraints of the feature model is required. We further plan to explore how research from the SPL community can better support the evolution of custom-developed variability representations. In parallel, we investigate possibilities to partly automate the PG's creation, e.g., from CAD product drawings, to reduce the manual effort for researchers and basic engineers.

## ACKNOWLEDGMENTS

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.



## REFERENCES

[1] Mathieu. Acher, Roberto Lopez-Herrejon, and Rick Rabiser. 2017. Teaching Software Product Lines: A Snapshot of Current Practices and Challenges. *ACM Transactions on Computing Education* 18, 1 (2017), 2:1–2:31.

[2] Wesley K. G. Assunção, Roberto E. Lopez-Herrejon, Lukas Linsbauer, Silvia R. Vergilio, and Alexander Egyed. 2017. Reengineering legacy applications into software product lines: a systematic mapping. *Empirical Software Engineering* 22, 6 (01 Dec 2017), 2972–3016.

[3] Noor Hasrina Bakar, Zarinah M. Kasirun, and Norsaremah Salleh. 2015. Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. *Journal of Systems and Software* 106 (2015), 132–149.

[4] Rabih Bashroush, Muhammad Garba, Rick Rabiser, Iris Groher, and Goetz Botterweck. 2017. CASE Tool Support for Variability Management in Software Product Lines. *ACM Computing Surveys (CSUR)* 50, 1 (2017), 14:1–14:45.

[5] Don Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *Software Product Lines*, Henk Obbink and Klaus Pohl (Eds.). Springer, 7–20.

[6] Guillaume Bécan, Nicolas Sannier, Mathieu Acher, Olivier Barais, Arnaud Blouin, and Benoit Baudry. 2014. Automating the formalization of product comparison matrices. In *ASE*. ACM, 433–444.

[7] Maurice H. ter Beek, Klaus Schmid, and Holger Eichelberger. 2019. Textual Variability Modeling Languages: An Overview and Considerations. In *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B (SPLC '19)*. Association for Computing Machinery, New York, NY, USA, 151–157.

[8] Luca Berardinelli, Alexandra Mazak, Oliver Alt, Manuel Wimmer, and Gerti Kappel. 2017. Model-driven systems engineering: Principles and application in the CPPS domain. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 261–299.

[9] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wąsowski. 2013. A survey of variability modeling in industrial practice. In *Proc. of the 7th Int. Workshop on Variability Modelling of Software-intensive Systems*. ACM, 7–14.

[10] Stefan Biffl, Detlef Gerhard, and Arndt Lüder. 2017. Introduction to the Multi-Disciplinary Engineering for Cyber-Physical Production Systems. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 1–24.

[11] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. 2017. Model-driven software engineering in practice. *Synthesis Lectures on Soft. Eng.* 3, 1 (2017), 1–207.

[12] Lianping Chen and Muhammad Ali Babar. 2011. A systematic review of evaluation of variability management approaches in software product lines. *IST* 53, 4 (2011), 344–362.

[13] Daniel Cruz, Eduardo Figueiredo, and Jabier Martinez. 2019. A Literature Review and Comparison of Three Feature Location Techniques Using ArgoUML-SPL. In *Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems (VAMOS '19)*. ACM, New York, NY, USA, 16:1–16:10.

[14] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. 2012. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. In *Proc. of the 6th International Workshop on Variability Modeling of Software-Intensive Systems*. ACM, 173–182.

[15] Krzysztof Czarnecki and Andrzej Wąsowski. 2007. Feature Diagrams and Logics: There and Back Again. In *Proc. of the 11th International Software Product Line Conference*. IEEE, 23–34.

[16] Deepak Dhungana, Paul Grünbacher, and Rick Rabiser. 2011. The DOPLER Meta-Tool for Decision-Oriented Variability Modeling: A Multiple Case Study. *Automated Software Engineering* 18, 1 (2011), 77–114.

[17] Sascha El-Sharkawy, Stephan Dederichs, and Klaus Schmid. 2012. From Feature Models to Decision Models and Back Again: An Analysis Based on Formal Transformations. In *Proc. of the 16th International Software Product Line Conference*. ACM, 126–135.

[18] Kevin Feichtinger and Rick Rabiser. 2020. Variability Model Transformations: Towards Unifying Variability Modeling. In *Proc. of the 46th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, Portoroz, Slovenia.

[19] Matthias Galster, Danny Weyns, Dan Tofan, Bartosz Michalik, and Paris Avgeriou. 2013. Variability in software systems-a systematic literature review. *IEEE Transactions on Software Engineering* 40, 3 (2013), 282–306.

[20] Øystein Haugen, Andrzej Wąsowski, and Krzysztof Czarnecki. 2013. CVL: common variability language. In *Proc. of the 17th International Software Product Line Conference*. ACM, 277–277.

[21] Didac Gil De La Iglesia and Danny Weyns. 2015. MAPE-K Formal Templates to Rigorously Design Behaviors for Self-Adaptive Systems. *ACM Trans. Auton. Adapt. Syst.* 10, 3, Article 15 (Sept. 2015), 31 pages.

[22] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson. 1990. *Feature-oriented domain analysis (FODA) feasibility study*. Technical Report. Carnegie-Mellon Univ., Pittsburgh, Pa, Software Engineering Inst.

[23] Sebastian Krieter, Thomas Thüm, Sandro Schulze, Gunter Saake, and Thomas Leich. 2020. YASA: Yet Another Sampling Algorithm. In *Proc. of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems*. ACM, 1–10.

[24] Lukas Linsbauer, Roberto Erick Lopez-Herrejon, and Alexander Egyed. 2014. Feature Model Synthesis with Genetic Programming. In *Search-Based Software Engineering*, Claire Le Goues and Shin Yoo (Eds.). Springer International Publishing, Cham, 153–167.

[25] Jens Meinicke, Thomas Thüm, Reimar Schröter, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2017. *Mastering Software Variability with FeatureIDE*. Springer.

[26] Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2019. Towards Modeling Variability of Products, Processes and Resources in Cyber-Physical Production Systems Engineering. In *SPLC (B)*. ACM, 68:1–68:8.

[27] Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2020. Feature Identification for Engineering Model Variants in Cyber-Physical Production Systems Engineering. In *VaMoS*. ACM, 18:1–18:5.

[28] László Monostori. 2014. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP* 17 (2014), 9–13.

[29] Sana Ben Nasr, Guillaume Bécan, Mathieu Acher, João Bosco Ferreira Filho, Nicolas Sannier, Benoit Baudry, and Jean-Marc Davril. 2017. Automated extraction of product comparison matrices from informal product descriptions. *J. Syst. Softw.* 124 (2017), 82–103.

[30] Sebastian Oster, Florian Markert, and Philipp Ritter. 2010. Automated Incremental Pairwise Testing of Software Product Lines. In *Software Product Lines: Going Beyond*, Jan Bosch and Jaejoon Lee (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 196–210.

[31] Gilles Perrouin, Sagar Sen, Jacques Klein, Benoit Baudry, and Yves le Traon. 2010. Automated and Scalable T-wise Test Case Generation Strategies for Software Product Lines. In *2010 Third International Conference on Software Testing, Verification and Validation*. 459–468.

[32] Klaus Pohl, Günter Böckle, and Frank J van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer Science & Business Media.

[33] Mikko Raatikainen, Juha Tiihonen, and Tomi Männistö. 2019. Software product lines and variability modeling: A tertiary study. *Journal of Systems and Software* 149 (2019), 485–510.

[34] Rick Rabiser, Klaus Schmid, Martin Becker, Goetz Botterweck, Matthias Galster, Iris Groher, and Danny Weyns. 2019. Industrial and Academic Software Product Line Research at SPLC: Perceptions of the Community. In *Proceedings of the 23rd International Systems and Software Product Line Conference*. ACM, Paris, France, 189–194.

[35] Fabricia Roos Frantz, David Felipe Benavides Cuevas, and Antonio Ruiz Cortés. 2009. Feature model to orthogonal variability model transformation towards interoperability between tools. In *Knowledge Industry Survival Strategy Initiative, Kiss Workshop (ASE 2009), Auckland, New Zealand*.

[36] Nicolas Sannier, Guillaume Bécan, Mathieu Acher, Sana Ben Nasr, and Benoit Baudry. 2014. Comparing or configuring products: are we getting the right ones?. In *VaMoS*. ACM, 9:1–9:7.

[37] Nicolas Sannier, Guillaume Bécan, Sana Ben Nasr, and Benoit Baudry. 2013. On Product Comparison Matrices and Variability Models from a Product Comparison/Configuration Perspective.

[38] Miriam Schleipen, Arndt Lüder, Olaf Sauer, Holger Flatt, and Jürgen Jasperneite. 2015. Requirements and concept for Plug-and-Work. *at-Automatisierungstechnik* 63, 10 (2015), 801–820.

[39] Pierre-Yves Schobbens, Patrick Heymans, and Jean-Christophe Trigaux. 2006. Feature diagrams: A survey and a formal semantics. In *Proc. of the 14th IEEE International Requirements Engineering Conference*. IEEE, 139–148.

[40] Thomas Thüm, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake. 2014. A Classification and Survey of Analysis Strategies for Software Product Lines. *ACM Comput. Surv.* 47, 1, Article 6 (June 2014), 45 pages.

[41] Frank van der Linden, Klaus Schmid, and Eelco Rommes. 2007. *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering*. Springer Berlin Heidelberg.

### 5.3.2 A reusable set of real-world product line case studies

#### Citation

[127] **K. Meixner**, K. Feichtinger, R. Rabiser, and S. Biffl. A reusable set of real-world product line case studies for comparing variability models in research and practice. In M. Mousavi, P.-Y. Schobbens, H. Araujo, I. Schaefer, M. ter Beek, X. Devroey, J. Rojas, R. Rabiser, M. Varshosaz, T. Kishi, and J. Lee, editors, *SPLC '21: 25th ACM International Systems and Software Product Line Conference, Leicester, United Kindom, September 6-11, 2021, Volume B*, volume Part F171625-B, pages 105–112. Association for Computing Machinery, 2021. doi: 10.1145/3461002.3473946

#### Aim

The publication “A reusable set of real-world product line case studies for comparing variability models in research and practice” [127] provides four real-world case studies with their product, process, and resource variability as a foundation for the evaluation of the approaches and techniques presented in this thesis. formalized in the PPR–DSL The publication (i) describes four real-world case studies of different complexity, (ii) formalizes then in the PPR–DSL, and (iii) introduces the VERT approach to transform the variability modeled in the industrial engineering artifact into state-of-the-art variability models, i.e., feature models.

#### Contribution to the thesis

This publication contributes to the research goals case studies for evaluation formalized in PPR–DSL (**G5**.), CPPS engineering artifacts with variability (**G5**.), and CPPS engineering artifact to variability model transformations (**G4**.).

This publication contributes to **RQ1**. and **RQ3**. by addressing the VDI 3695 measures of *models and description languages* M1., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of interdisciplinary collaboration* C5..

#### Abstract

Real-world cases describing (product) variability in production systems are rare and often not accessible. Thus, researchers often use toy examples or develop fictitious case studies. These are designed to demonstrate their approach but rarely to compare multiple approaches. In this paper, we aim at making variability modeling evaluations comparable. We present and provide a reusable set of four real-world case studies that are easy to access, with artifacts represented in a universal, variability-model-agnostic way, the industrial PPR–DSL. We report how researchers can use the case studies, automatically transforming the DSL artifacts to well-known variability models, e.g., product feature models, using the VERT process. We compare the expressiveness and complexity of the

transformed feature models. We argue that the case studies with the DSL and the flexible transformation capabilities build a valuable contribution to making future research results more comparable and facilitating evaluations with real-world product lines.

# A Reusable Set of Real-World Product Line Case Studies for Comparing Variability Models in Research and Practice

Kristof Meixner
CDL SQI, ISE
TU Wien, Austria
kristof.meixner@tuwien.ac.at

Kevin Feichtinger
LIT CPS Lab
JKU Linz, Austria
kevin.feichtinger@jku.at

Rick Rabiser
CDL VaSiCS, LIT CPS Lab
JKU Linz, Austria
rick.rabiser@jku.at

Stefan Biffl
Inf. & Software Eng.
TU Wien and CDP, Austria
stefan.biffl@tuwien.ac.at

## ABSTRACT

Real-world cases describing (product) variability in production systems are rare and often not accessible. Thus, researchers often use toy examples or develop fictitious case studies. These are designed to demonstrate their approach but rarely to compare multiple approaches. In this paper, we aim at making variability modeling evaluations comparable. We present and provide a reusable set of four real-world case studies that are easy to access, with artifacts represented in a universal, variability-model-agnostic way, the industrial Product-Process-Resource Domain-Specific Language (PPR DSL). We report how researchers can use the case studies, automatically transforming the Domain-Specific Language (DSL) artifacts to well-known variability models, e.g., product feature models, using the Variability Evolution Roundtrip Transformation (VERT) process. We compare the expressiveness and complexity of the transformed feature models. We argue that the case studies with the DSL and the flexible transformation capabilities build a valuable contribution to making future research results more comparable and facilitating evaluations with real-world product lines.

## CCS CONCEPTS

• **Software and its engineering** → **Software product lines**.

## KEYWORDS

Variability Modeling, Feature Extraction, Cyber-Physical Production System, Case Studies.

**ACM Reference Format:**
Kristof Meixner, Kevin Feichtinger, Rick Rabiser, and Stefan Biffl. 2021. A Reusable Set of Real-World Product Line Case Studies for Comparing Variability Models in Research and Practice. In *25th ACM International Systems and Software Product Line Conference - Volume B (SPLC '21), September 6–11, 2021, Leicester, United Kingdom.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3461002.3473946

## 1 INTRODUCTION

In Software Product Line (SPL) engineering developers aim at developing software(-intensive) systems through systematic reuse of artifacts and their customization [4]. SPL engineering uses variability modeling to explicitly represent the commonalities and variability of reusable artifacts and their dependencies for building and evolving the family of underlying systems [1].

In Cyber-Physical Production Systems (CPPSs) engineering, engineers from different disciplines develop software-intensive systems that manufacture product lines employing modern production techniques [3]. CPPSs can adapt to uncertain conditions of their physical environment using the latest information and communication technology [18]. Designing a CPPS requires variability modeling on multiple levels, e.g., the product, process, production resource, and software level. Therefore, engineers create different engineering artifacts (e.g., product comparison matrices (PCMs) [19] or CAD drawings ) that contain variant information. These artifacts are often unstructured and need to be analyzed to model and extract the overall CPPS variability. While the amount of structured variability modeling approaches is overwhelming, industrial practitioners are often unaware of available approaches and their application [2, 20].

Case studies can help researchers and practitioners to gain insights into variability modeling in different contexts. They are also a major empirical strategy to describe and investigate phenomena in software engineering [22]. Examples in SPL engineering are, e.g., the challenge repository of the SPL community[1], the SPL2go [23] catalog of SPLs, the SPLOT repository [17], the Extractive Software Product Line Adoption (ESPLA) case study catalog [13], and the Apo-Games [12] case study on reverse SPL engineering.

However, case studies for extractive variability modeling are often not fully accessible nor easy to reproduce [13]. According to the ESPLA catalog about 20 case studies in the catalog describe variability in production systems engineering, but none of them are accessible. On variable production systems, we found a single accessible case study only, the *Pick-and-Place Unit (PPU)* [24].

This shortage tempts researchers to often use toy examples or fictitious case studies to demonstrate their particular approaches [13]. However, such case studies are often not reusable in different contexts or for different approaches. This makes it challenging to compare variability modeling approaches and their evaluations for potential use in the industry. Hence, we raise the question, *which set of real-world cases can be used to investigate variability modeling approaches* and *how can these cases be used in research and practice?* The effort to manually create (and maintain) variability models from industrial artifacts is very high [1]. Therefore, we also need to investigate how to efficiently extract variability information from engineering artifacts and create state-of-the-art variability models.

This work aims to make evaluations for CPPSs easier to reproduce and more comparable. Hence, this paper introduces a reusable

[1]https://variability-challenges.github.io/

set of four real-world case studies for product lines in CPPS engineering: (i) the 3D-printed truck, (ii) the shift fork, (iii) the water filter, and (iv) the rocker switch product lines, exhibiting varying levels of complexity from three domains.

The case studies are represented universally in the PPR DSL [16] as CPPS engineering artifact. The PPR DSL was created to represent the functional view on CPPSs including system variants. This allows to represent the required concepts for a transformation to different well-known variability models, e.g., feature models. Researchers can transform the PPR DSL instances into variability models to evaluate selected SPL approaches. For instance, this paper demonstrates how PPR DSL artifacts can be transformed into product feature models, using the *VERT* process [6]. We discuss the case studies and compare the resulting artifacts and models to investigate their expressiveness and complexity. The case studies can and should be used by the community to evaluate and compare innovative variability modeling approaches. DSL artifacts, feature models, and case descriptions are available in an online repository[2].

## 2 RESEARCH QUESTIONS

CPPS variability originates from different sources [21]. The characteristics and variability of the products manufactured in these systems influence all subsequent engineering activities and decisions. We aim to search for and elicit available case studies for product lines in production systems of different complexity, e.g., as in the number of product variants, their features, and constraints, as a foundation for investigating product variability in CPPS. We also investigate the applicability of our real-world case study systems to allow researchers to compare different variability modeling approaches. Therefore, we define the following research questions.

***RQ1.*** *Which real-world case studies satisfy minimal requirements as a basis to investigate product variability in production systems?* We propose minimal requirements for real-world case studies to investigate variability in CPPS. Adhering to these requirements, we selected and elicited four real-world case studies of product lines for production systems from ongoing collaboration projects. We present and describe the case study systems in Section 5.

***RQ2.*** *How can we obtain variability models from the real-world case study systems to compare different variability modeling approaches?* We show that our case studies can serve as a basis to investigate different variability modeling approaches and make them better comparable. To demonstrate this, we first describe the CPPS product variants using the industrial PPR DSL. We demonstrate how we transform the DSL instances of the real-word case study systems to feature models and back using the VERT process [6], which builds on the variability model transformation approach TRAVART [7]. We then compare these resulting models to the case study descriptions focusing on their features and constraints.

```
1  Attribute "length": { type: "Number", unit: "mm" }
2
3  Product "Chassis": { name: "Chassis" }
4  Product "Cabin": { name: "Cabin" }
5
6  Product "Body": { name: "Body",
7    isAbstract: true }
8
9  Product "Tank": { name: "Tank",
10   isAbstract: false ,
11   implements: [ "Body" ] }
12
13 Product "OpenTop": { name: "OpenTop",
14   isAbstract: false ,
15   implements: [ "Body" ], length: 30 }
16
17 Product "Legotruck": { name: "Legotruck",
18   isAbstract: true ,
19   children: [ "Chassis", "Cabin", "Body"],
20   requires: [ "Chassis", "Cabin", "Body"] }
21
22 Product "Legotruck1": { name: "Legotruck1",
23   isAbstract: false ,
24   implements: [ "Legotruck" ],
25   requires: [ "Tank" ], excludes: [ "OpenTop" ] }
```

**Listing 1: PPR DSL [16] excerpt of the *3D-printed truck.***

## 3 CPPS MODELING WITH VARIABILITY

To plan a CPPS, engineers first design its functional model using Product-Process-Resource (PPR) concepts including different variants. Therefore, Meixner et al. [16] designed the ***PPR DSL*** building on extensions of the Formalised Process Description (FPD) [9]. Product design (with variants) is a crucial part of CPPS planning as the variants and parameters determine the production processes and resources. For instance, a product *3D-printed Truck* (cf. Section 5.1) basically consists of a body, a chassis, and cabin but has several variations like a tank body. Depending on the variant, it needs to be assembled differently requiring different production resources.

Listing 1 shows a PPR DSL excerpt of the truck. Line 1 defines a (global) *Attribute length*. Lines 3 & 4 define two *Products Chassis* and *Cabin*. Line 6 defines an abstract product *Body* (keyword: *isAbstract*). An abstract product represents either a virtual product, e.g., to group similar products part types like bodies, or an assembly group, a set of functional connected parts that result from a production step. Lines 8 & 10 define the product *Tank* as implementation of the body (*implements*). Lines 11 to 13 define the *OpenTop* product implementing the body with a length of 30 mm. The abstract product *Legotruck* (Lines 15-18) holds the parts (*children*) and the required products (*requires* – similarly there exists an *excludes* keyword). We use the two product keywords *requires* and *excludes* to map dependencies and exclusion criteria of products. Abstract products can contain abstract and concrete products in these lists. Finally, the variant *Legotruck1* (Lines 20-24) implements the *Legotruck*, requiring the *Tank* as implementation for the *Body*. Furthermore, the product excludes the *OpenTop* as a possible body alternative. Similarly, the PPR DSL can describe production processes with their input and output products and resources.

However, for a more systematic management of variability, engineers would benefit from a dedicated variability model. Feature models are well-known to model the variability of different systems [5]. They can also represent the product variability modeled in the PPR DSL, which, however, requires a sufficient transformation. Figure 1 shows a feature model for the product described by the PPR DSL in Listing 1. Instead of representing the particular truck variants, it shows the features that the product line consists of. Due to space constraints and their prominence we refer to the relevant literature for details on feature models [14].

[2]https://github.com/tuw-qse/cpps-var-case-studies

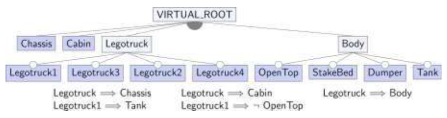


**Figure 1: Feature model of the *3D-printed truck*.**

## 4 STUDY REQUIREMENTS AND PROCESS

We identified the following requirements from discussions with researchers and practitioners in CPPS research and engineering and based on our long-time research experience in the field.

*Req1. Product variability in production systems.* The case study must cover the variability of products that can be manufactured on a production system. *Req1.* is important as this product variability, introducing configuration options that are non-Boolean and can be instantiated multiple times, drives the subsequent CPPS variability. The production process must allow for automation, at least in a future CPPS. Therefore, (i) we exclude software products and (ii) we leave process and resource variability of CPPSs for future work.

*Req2. Structured product variants.* The products produced in the CPPS need to be sufficiently similar to build at least a minimal product line. Product line engineering text books [4] argue that 50% to 80% commonalities are required for a product line. We would not want to specify an exact number but exclude case studies on special production or specialized manual labor, where products are unique with few commonalities.

*Req3. Availability of domain experts or documents.* This requirement is crucial to properly describe the particular case study. It requires domain experts who sufficiently understand the product line and a candidate production system to discuss variability and provide feedback. Further, the case study elicitation requires sufficient documentation, e.g., CAD drawings of product variants, to extract product variability information.

Based on these requirements, we performed an iterative research process to elicit the real-world case studies from our collaborations using the following three steps.

***1. Identify accessible real-world case studies.*** We employed strategies from two methodologies, the case study guideline [22] and the design science methodology [26]. Specifically, we conducted offline and online *interviews* with practitioners and researchers from three collaborations with industry and academia on production systems. Based on the discussions, we derived and defined minimal requirements on real-world case study systems that should be used to investigate variability in production systems. We describe these essential requirements in Section 5. From these collaborations, we identified and selected four particular cases that fulfill the requirements and gathered documentation material to study them.

***2. Extract variability information to PPR DSL.*** CPPS variability aspects are often described in various engineering artifacts [21]. Manually creating and maintaining variability models from such (industrial) artifacts is tedious [2]. Following the case study guideline, in a *data analysis* step, we modeled the product lines in the PPR DSL, checked the results with the interviewees, and *interpreted* the data. In the modeling process, we focused on the product variability of the case studies and left the CPPS's process and resource variability for future work. Therefore, the author that developed the DSL in close cooperation with industry modeled our four case study systems using the PPR DSL. The author mapped the product parts, the intermediate assembly groups, and the product variants to concepts in the DSL. The author also analyzed and mapped the dependencies between the products and between their parts to *is-part-of* (*children*), *requires*, or *excludes* concepts.

***3. Transform a PPR DSL artifact to a well-known variability model.*** Applying the VERT process [6] enables users to transform engineering artifacts containing variability information, such as the PPR DSL, into variability models like feature or decision models. Following the iterative design science methodology [26] one author (different from the author performing *Step 2*) defined a mapping between the PPR DSL and FeatureIDE [14] feature models. The mapping facilitated defining *transformation operations* between the two representations and implementing them as part of the TRAVART [7] approach. In Section 3, we describe the mapping between the PPR DSL and feature models. Additionally, the author sampled the configurations for each case study's resulting feature model using the YASA sampler [11] to estimate the configuration space for the case studies.

The ***VERT process*** used in *Step 3* builds on the variability artifact transformation approach TRAVART [7]. In this approach, *transformation operations* describe the operations required to transform one (type of) variability model into a different one (e.g, feature models into decision models), based on their meta-models. If no variability meta-model is available (e.g., for industry representations), these transformation operations define a mapping between the industry engineering artifact and the well-known variability model. *Transformation algorithms* implement the transformation operations between two concrete variability model types (e.g., feature model to decision model) or the industry representation to a variability model (e.g., the PPR DSL [16] to a feature model)

Based on learnings from earlier work [6], we map all product variability concepts of the PPR DSL to concepts of feature models. We accommodate for specific concepts of the PPR DSL, which a feature model cannot represent, e.g., custom attribute definitions or constraints based on these custom attributes, by storing them in feature properties. Table 1 shows the mapping in between the DSL artifact components and a feature model. The mappings are implemented so that the relevant information to restore the source model (type) are maintained. A detailed description of transformation operations is available online[2].

**Table 1: Mapping of PPR DSL to Feature Model.**

| | PPR DSL | Feature Model |
|---|---|---|
| *custom attribute* | attribute | property |
| | type | property |
| | unit | property |
| | defaultValue | property |
| | description | property |
| *product* | id | name |
| | name | property |
| | isAbstract | defines if the feature is abstract |
| | implements | property/feature tree |
| | requires | implies constraint |
| | excludes | excludes constraint |

## 5 CASE STUDIES

Based on the requirements we selected four case studies of real-world product lines in discrete manufacturing of different complexity. For each case study, we evaluate the VERT transformations by comparing the PPR DSL engineering artifacts and the feature models resulting from the transformations. Also, for each case study, we describe possible (and partly ongoing) applications in research and practice. Table 2 shows an overview comparing key characteristics of the case studies. The PPR DSL files, feature models, and additional material are available in the corresponding repository[2].

### 5.1 Case Study 3D-printed Truck

*Overview.* The real-world *3D-printed truck* product line is produced in an academic production system, the *Testbed for Industry 4.0 (I4.0)* at Czech Technical University in Prague[3]. The I4.0 Testbed is used for research and technology transfer on various topics of CPPSs and collaborative robot-human production, like process scheduling and optimization and adaptive production. The I4.0 Testbed can manipulate different products using three robot arms and a conveyor belt that can be flexibly configured. The truck case is of low complexity with just four possible types and no dependencies between the features in the product structure itself.

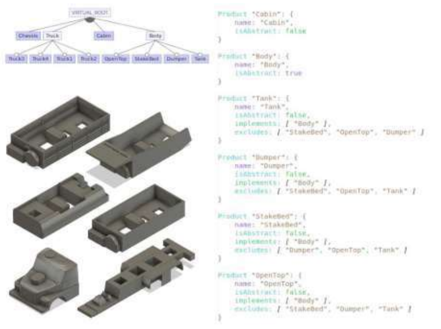
*Data collection.* We interviewed one researcher, with whom we have a long-running collaboration, at online and offline meetings, i.a., during a visit at the I4.0 Testbed, in semi-structured interviews. Further, we elicited relevant parts of the case study from documents that we received from the researcher, including publications [25].

*Variability.* To research the mentioned phenomena, the I4.0 Testbed is configured to assemble the *truck product line*, currently consisting of four different 3D-printed truck variants. Basically, a truck consists of three basic parts, the *chassis*, the *cabin*, and the *body*. While the chassis and the cabin are the same in all truck variants, there are four types of truck bodies: *dumper*, *opentop*, *stakebed*, and *tank*. In addition, the single parts of the truck can be produced in different colors. Wally et al. [25] discussed the I4.0 Testbed as a running use case for iterative process refinement, also providing a feature model of the truck product line. We do not consider the colors as specific features to make the truck product line comparable to the other case studies in this paper, where we also just consider parts rather than their particular attributes.

*Configuration/Assembly.* When assembling the truck, the sequence of production steps and the production resources depend on the particular truck configuration. For instance, when assembling the dumper configuration, the cabin needs to be mounted to the chassis before the body. Similarly, the tank variant requires a production resource that is narrow enough so it can grip between two particular points.

*Representing Variability using the PPR DSL.* An author of the paper modeled the 3D-printed truck product line of the case study in the PPR DSL. The four product variants of the product line require twelve product definitions in the DSL, e.g., *body* (Product) at Line 6, including two abstract products (isAbstract) as templates for concrete products. The products *tank* and *opentop* then implement the template of the *body* (implements). The instance defines a *truck* to consist of (children) and to require (requires) the concrete product

[3]Industry 4.0 Testbed: https://ciirc.cvut.cz/teams-labs/testbed/



```
Product "Cabin": {
    name: "Cabin",
    isAbstract: false
}
Product "Body": {
    name: "Body",
    isAbstract: true
}
Product "Tank": {
    name: "Tank",
    isAbstract: false,
    implements: [ "Body" ],
    excludes: [ "StakeBed", "OpenTop", "Dumper" ]
}
Product "Dumper": {
    name: "Dumper",
    isAbstract: false,
    implements: [ "Body" ],
    excludes: [ "StakeBed", "OpenTop", "Tank" ]
}
Product "StakeBed": {
    name: "StakeBed",
    isAbstract: false,
    implements: [ "Body" ],
    excludes: [ "Dumper", "OpenTop", "Tank" ]
}
Product "OpenTop": {
    name: "OpenTop",
    isAbstract: false,
    implements: [ "Body" ],
    excludes: [ "StakeBed", "Dumper", "Tank" ]
}
```

**Figure 2: Rendered 3D-printed truck (©Vaclav Jirkovsky in [25]), DSL section, and the resulting feature model.**

parts *chassis* and *cabin* and an abstract *body*. The concrete product *truck1* implements (implements) the truck and requires the concrete body implementation *tank*. It excludes the *opentop* product part with the excludes keyword as only one of the truck bodies can be used at a time. In total, 31 requires and excludes dependencies define the product variants (cf. Table 2).

*Transformed Variability Models.* The VERT transformation of the PPR DSL instance to a feature model resulted in a model with 13 features[4] and 31 constraints. Figure 1 shows the resulting feature model with five example constraints. We sampled 23 configurations for this case study.

*Applications and Summary.* Figure 2 shows the parts of the 3D-printed truck as rendering with a section of the corresponding PPR DSL instance and the resulting feature model. The complexity of the 3D-printed truck product line with just four product variants, two abstract assembly groups that categorize subtree, and 30 dependencies (e.g., exclusion criteria between the body parts) between the products is quite low. However, the case is a simple and straightforward example to learn about/teach variability modeling concepts. The DSL instance and the resulting feature model can be found in the repository[5].

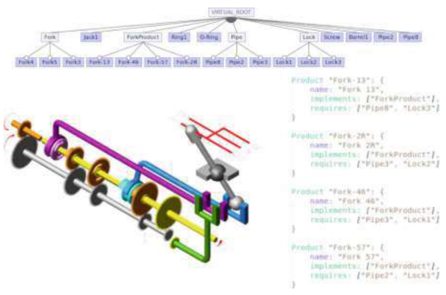
### 5.2 Case Study Shift Fork

*Overview.* The *shift fork* product line (cf. also Feichtinger et al. [6]) is produced on an industrial production system planned by one of our industry partners, a CPPS integrator for high-speed automation of assembling automotive parts. The data reported (artifacts and examples) has been abstracted to honor intellectual property rights. Currently, our industry partner restructures its engineering process towards using the PPR approach and an approach for structured product line engineering through all engineering levels. To this end, we are in close contact with two senior and one junior domain expert of the company that act as an interface to research.

[4]The additional feature originates from the root node of the feature model, which was counted towards the number of overall features.
[5]https://github.com/tuw-qse/cpps-var-case-studies/tree/main/truck

**Table 2: Overview of the characteristics of the four case studies.**

| Parts/Types | Complexity | #Product Variants | #DSL Elements | #Dependencies | #Features | #Constraints | #Configurations |
|---|---|---|---|---|---|---|---|
| 3D-printed truck | low | 4 | 12 | 31 | 13 | 31 | 23 |
| Shift fork | low | 4 | 22 | 36 | 25 | 38 | 67 |
| Water filter | medium | 8 | 54 | 165 | 55 | 165 | 217 |
| Rocker switch | medium | 12 | 54 | 184 | 55 | 216 | 183 |



```
Product "Fork-13": {
    name: "Fork 13",
    implements: ["ForkProduct"],
    requires: ["Pipe8", "Lock3"]
}

Product "Fork-2R": {
    name: "Fork 2R",
    implements: ["ForkProduct"],
    requires: ["Pipe3", "Lock2"]
}

Product "Fork-46": {
    name: "Fork 46",
    implements: ["ForkProduct"],
    requires: ["Pipe3", "Lock1"]
}

Product "Fork-57": {
    name: "Fork 57",
    implements: ["ForkProduct"],
    requires: ["Pipe2", "Lock1"]
}
```

**Figure 3: Rendered manual transmission with shift forks (©World Arm Lamp, CC0 on Wikipedia - https://w.wiki/3DCf), DSL section, and the resulting feature model.**

*Data collection.* For the case study, we conducted several offline and online workshops. In these workshops, we discussed the case in detail with the two mentioned senior members, two senior engineers (one from the functional and one from the mechanical domain), and several junior engineers from different domains. We received and discussed a large number of different engineering artifacts, from requirement documents in text and spreadsheet format to the manual of the finished production system. We analyzed the data manually, extracting the information of the product line and its variants to derive the case study.

*Variability.* A shift fork (cf. schematic view in Figure 3) is a part of a manual transmission in the drive chain of a machine, e.g., a car. It shifts a cuff along rods into particular positions to let the gears connect correctly. The *shift fork product line* consists of four different shift fork variants manufactured on a timed conveyor-belt production system.

A single shift fork consists of 14 parts, where the same 12 parts are used in all four shift fork variants, and two parts differ between the variants. For instance, the shift fork variant *Fork-13*[6] requires *Barrel 1*, a commonality of all shift fork types, as well as *Pipe 8* and *Lock 3*, which differ among the shift fork variants. As there are, e.g., different pipe types, but a shift fork requires exactly one pipe, the pipe types build an exclusive group. The production system requires 15 process steps to assemble a single shift fork.

*Configuration/Assembly.* Similar to the truck, the sequence of process steps, i.e., how the shift fork is assembled, is important in the production system. For instance, the parts *Fork 3*, *Fork 4*, and *Fork 5* need to be mounted to the pipe before one of the locks can be welded on. Such dependencies cannot be expressed, for example, in a feature model, since this model does not define the order of feature selection (which requires additional approaches like decision models).

*Representing Variability using the PPR DSL.* The PPR DSL model of the case study, created by an author, consists of 22 products, including four abstract products. The model defines 36 inclusion and exclusion dependencies between the products and their parts.

*Transformed Variability Models.* The feature model resulting from transforming the shift fork PPR DSL instance contains 13 features, 31 constraints, and 23 configurations sampled by YASA. For the resulting feature model we refer to the online repository due to readability and space constraints.

*Applications and Summary.* Figure 3 depicts a rendered manual transmission with three shift forks (blue, cyan, and green), the corresponding PPR DSL instance for the shift fork product types, and the resulting feature model. The complexity of the shift fork case with four products, four abstract assembly groups, and 36 dependencies (e.g., exclusions between the pipes and locks) between the product parts is rather low. The DSL instance and the resulting feature model can be found in the repository[6]. Our industry partner uses the shift fork case to investigate variability in production systems. This includes basic teaching of variability models, like feature models, and examining different techniques to introduce structured variability modeling into their CPPS engineering process.

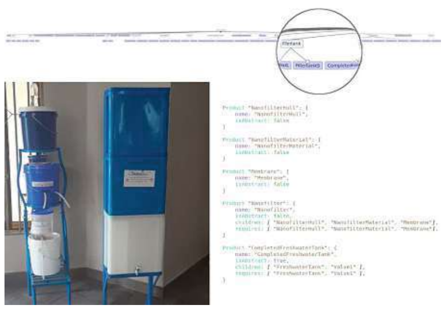
## 5.3 Case Study Water Filter – NanoFilter

*Overview.* The *water filter* product line, the so-called *NanoFilter* (cf. Figure 4), originates from a development project in Tanzania[7]. In the project, researchers investigated different filter materials and constructions [10]. The water filter can filter impurities and, contrary to most local solutions, selectively remove contaminants from unsafe water sources. Depending on its configuration, the *NanoFilter* costs about 5% to 25% less than imported water filters. This way, the water filter addresses the basic need of price-sensitive customers in developing countries by providing a low-cost but customizable solution that works without electricity and can be manufactured locally due to its simple structure and few components.

*Data Collection.* A collaborating researcher (not an author of this paper) collected the primary data for the case study by conducting a semi-structured online interview and email communication with the *Nanofilter*'s inventor. This data has been supported by secondary data from the filter's manual, project documents submitted for funding[8], and the company's website. We iteratively elicited and

[6]https://github.com/tuw-qse/cpps-var-case-studies/tree/main/shiftfork

[7]See project website https://gongalimodel.com/
[8]www.globalgiving.com



```
Product "NanofilterHull": {
    name: "NanofilterHull",
    isAbstract: false
}
Product "NanofilterMaterial": {
    name: "NanofilterMaterial",
    isAbstract: false
}
Product "Membrane": {
    name: "Membrane",
    isAbstract: false
}
Product "Nanofilter": {
    name: "Nanofilter",
    isAbstract: false,
    children: [ "NanofilterHull", "NanofilterMaterial", "Membrane"],
    requires: [ "NanofilterHull", "NanofilterMaterial", "Membrane"],
}
Product "CompletedFreshwaterTank": {
    name: "CompletedFreshwaterTank",
    isAbstract: true,
    children: [ "FreshwaterTank", "Valve1" ],
    requires: [ "FreshwaterTank", "Valve1" ],
}
```

**Figure 4: NanoFilter installed on a Rack (©Askwar Hilonga, [8]), DSL section, and the resulting feature model.**

reviewed the case study from the gathered data and the resulting knowledge from both data sources.

*Variability.* The *water filter* product line consists of eight variants that contain up to 14 parts. To meet particular requirements, e.g., use at home vs. in schools, or the water contamination rate, the water filter variants differ in size, clean water production rate, and the water purification technology. The water filter consists of a standard or large *freshwater tank* for the cleaned water and a *filter tank* of similar dimensions to purify the water. The *filter tank* always contains *sand* and either *bone charcoal* or *active charcoal* to filter particles and micro-organisms from the polluted water. For severe water contamination, like certain chemicals, the water filter can have an additional *nano filter.* There is an additional *wastewater tank* to hold the polluted water for large variants of the water filter. Depending on the size of the water filter and whether the *nano filter* is installed, the water filter is mounted on different sorts of mounts like an *iron frame* or a large *rack*. Furthermore, some auxiliary parts, such as *valves* and *tubes*, need to be installed.

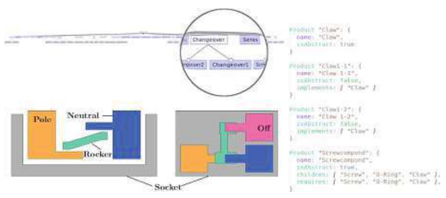
*Configuration/Assembly.* The assembly sequence of the water filter is relatively straightforward and implied by the configuration of the water filter. However, to assemble the different configurations, different resources might be needed, e.g., to lift the tanks and racks.

*Representing Variability using the PPR DSL.* An author of the paper created a model of the *water filter* product line in the PPR DSL. The DSL instance describes 54 products with ten abstract products. From the total products, 19 were created as assembly groups that build functional, connected intermediate products in the production process. Furthermore, we created 165 inclusion and exclusion dependencies.

*Transformed Variability Models.* The feature model resulting from transforming the PPR DSL instance of the water filter product line contains 55 features, 165 constraints, and 217 possible configurations. For the resulting feature model we refer to the online repository[9] due to readability and space constraints.

*Applications and Summary.* Figure 4 shows a picture of two Nanofilter water filters in different development stages with three tanks and a nanofilter in the left water filter. The figure also shows a section of the PPR DSL for the water filter product variants and the transformed feature model. The water filter case consists of eight product types and with 165 dependencies between the parts of the product types (e.g., exclusion of bone and active charcoal). Therefore, we consider the case's complexity to be at a medium level. The additional information of the water filter case studies can be found at this paper's repository. With the water filter case, we aim to investigate the evolution of variability. An interesting issue is how an iterative addition of product variants will impact the complexity of the variability model. Furthermore, production systems engineers want to know which product variants are likely to add over-proportional cost to the production system (design).

[9]https://github.com/tuw-qse/cpps-var-case-studies/tree/main/waterfilter



```
Product "Claw": {
  name: "Claw",
  isAbstract: true
}
Product "Claw1-1": {
  name: "Claw 1-1",
  isAbstract: false,
  implements: [ "Claw" ]
}
Product "Claw1-2": {
  name: "Claw 1-2",
  isAbstract: false,
  implements: [ "Claw" ]
}
Product "Screwcompound": {
  name: "Screwcompound",
  isAbstract: true,
  children: [ "Screw", "O-Ring", "Claw" ],
  requires: [ "Screw", "O-Ring", "Claw" ],
}
```

**Figure 5: Schematic view of a rocker switch variant with three contact types and one rocker type.**

### 5.4 Case Study Rocker Switch

*Overview.* The *rocker switch* product line (a simplified example was presented by Meixner et al. [15]) is produced on a high-speed industrial production system by the same industry partner who produces the shift fork (see above). The data reported (artifacts and examples) has been abstracted to honor intellectual property rights.

*Rocker switches* are everyday appliances to control electrical devices, such as lights or sun blinds. Figure 5 shows a schematic image of an assembly of a rocker switch variant. Basically, a *rocker switch* consists of several contacts and rockers that close and open electric circuits. With this simple schema, it is possible to realize basic but also complicated switching applications. The rocker switch production system is a fix-clocked conveyor belt system (cf. Section 5.2).

*Data Collection.* Similar to the shift fork case study, we conducted several offline and online workshops for the rocker switch case study with our industry partner. There we discussed the rocker switch product line and production system with the two main contact engineers and one senior functional planner. Furthermore, we received detailed planning artifacts, like a spreadsheet (~300 rows × 45 columns) that engineers use as variant matrices, which we incorporated into the case study. The engineers reviewed and acknowledged the elicited results and drawn conclusions.

*Variability.* The *rocker switch* product line consists of the essential core of twelve different rocker switches variants with up to 24 product parts and ten process steps (some variants in the original product line consisted of up to 35 parts and 60 process steps). Each rocker switch variant has an initially empty *socket* where parts are mounted. The rocker switch variant shown in Figure 5 has a socket with four spaces for contacts and contacts of type *pole*, *neutral*, and *off*. In the middle of the contacts, the figure shows a *rocker* that allows opening and closing the circuits. Depending on the switching application, the rocker switch consists of various contact and rocker types mounted at different locations in the socket. To mount the socket into a frame, each rocker switch has two *claws* with isolating *rings* each held by a *screw* (not shown in Figure 5).

*Configuration/Assembly.* The left-hand-side part of Figure 5 shows dependencies of the assembly sequence on the configuration. For instance, in this variant, the pole needs to be inserted into the socket before the rocker. Accordingly, the rocker must be inserted before the neutral and off contacts. However, in this case, it is not important whether the neutral or the off contact get mounted first or whether the claw is attached. Furthermore, some resources, like a robot arm, of the production system might be able to handle different contact types that can be used in different process steps when the steps are distinct between rocker switch variants. These variation points represent a large solution space for the design of the production system.

*Representing Variability using the PPR DSL.* The DSL instance of the *rocker switch* product line, which an author created, consists of 54 products with 15 abstract products. The instance further has four assembly groups and defines 184 inclusion and exclusion dependencies.

*Transformed Variability Models.* The feature model resulting from the transformation of the rocker switch PPR DSL contains 55 features, 216 constraints, and 183 possible configurations. For the resulting feature model we refer to the online repository[10] due to readability and space constraints.

*Applications and Summary.* Figure 5 shows a schematic view of a rocker switch variant, a part of its PPR DSL representation, and a section of the feature model resulting from the transformation. The rocker switch consists of 12 product variants with 165 dependencies, e.g., to define which contact groups are valid for specific product types, and 15 assembly groups. We consider the product line's complexity to be medium. The case study's additional material, like the DSL instance and the feature model, can be found in the repository. The rocker switch case is crucial for investigating further variability models together with our industry partner. The product configuration has a direct impact on the production step sequence. If two product types have distinct process steps with similar needs, the resources in the production system might be reused. For instance, if two different but similar contacts need to be inserted into a socket and two product variants use the process steps exclusively, a robot arm might be reused. As feature models do not imply an order for feature selection, we need to investigate for this case the transformation to other variability models, like decision models.

## 6 DISCUSSION

The variability in CPPSs is rooted in different aspects of the system, like products, processes, and resources. As these aspects are intertwined, research needs to take their combination into account, including the domain's particular challenges [21], which also requires a combination of variability modeling approaches. In this paper, we focus on product variability in CPPSs striving for real-world case studies that can be used in research and practice, following the research questions raised in Section 2.

To address ***RQ1***, in Section 5 we defined three essential requirements that case studies need to satisfy to be selected into our set. A real-world case study system to be considered must (*Req1.*) cover variability of products that can be produced on a CPPS, (*Req2.*) describe product variants that are sufficiently similar to build a product family, and *(Req3.)* must be accessible (either previously or published with the contribution).

In Section 5 we introduced four case studies on product variability from the production domain that satisfy the identified requirements and represent several complexity levels, i.e., regarding the number of product variants and constraints. The product variant data of the case studies are represented in the PPR DSL (i) to make them accessible in a uniform and readable way and (ii) as a foundation for deriving a variety of product variability models following on the VERT/TRAVART method. We provide the case studies in a repository[2] that contains the case descriptions, the DSL instances, their resulting feature models, and the transformation algorithms. This way, we want to make the case studies available for a broad range of researchers and practitioners.

With this set of reusable case studies, we address product lines in CPPS in contrast to case study directories like the SPL challenge or ApoGames [12] repository that cover SPLs only. While the ESPLA catalog [13] contains some product lines for production systems, they are not accessible. In the future, we also aim to contribute our case studies to the ESPLA catalog.

To address ***RQ2***, in Section 3 we introduced operations to transform the PPR DSL to a feature model. In Section 5, we described how we modeled the product line data of the case studies in the PPR DSL. Using the defined transformation operations, we automatically transformed the PPR DSL instances to feature models and compared the case study characteristics based on these models. In this context, the PPR DSL is sufficiently expressive to serve as the basis to transform the product variants to a suitable feature model. Furthermore, we describe examples of how the case studies can be used in research and practice. These research results build on [6, 16] and go beyond the state of the art in variability modeling by (i) uniformly describing the cases in a PPR DSL engineering artifact and (ii) facilitating the transformation to feature models and their comparison.

The PPR DSL allows the similar description of other CPPS product lines, ensuring the basic quality of case study descriptions. Due to using the PPR DSL and reusing the TRAVART transformation operations, the resulting variability models can be expected to be comparable among each other and of similar quality. Researchers, who want to represent the case studies' product lines in a particular variability model, can implement transformation operations for their approach to automatically transform the cases into their model. This way, they can design and implement comparable variability models in their research and application scopes.

These research results go beyond the state of the art in variability modeling by (i) providing transformation operations to transform PPR DSL engineering artifacts to feature models, and (ii) explaining how to design transformation operations to transform the PPR DSL to variability models other than feature models.

[10]https://github.com/tuw-qse/cpps-var-case-studies/tree/main/rockerswitch

**Threats to Validity.** Despite being real-world case studies, the case studies are still of limited size and complexity. Furthermore, the case studies have not been found or selected systematically, e.g., to cover particular production system fields. They were rather identified and selected from ongoing collaborations where we had access to people willing to contribute. The case studies were modeled in the DSL by a single author. Similarly, the transformation operations were implemented by a (different) single author. However, we tried to reduce bias by regularly checking results with a second author. The case studies solely describe product lines and not the complete variability of the production system, e.g., process variability. However, we aim at extending the case studies in this direction.

## 7 CONCLUSIONS AND FUTURE WORK

Unfortunately, real-world cases describing variability in production systems are rare and often not accessible. We identified selection criteria for real-world case studies of product lines in CPPS engineering and presented a reusable set of four case studies with varying complexity from three different domains. We modeled the product lines of the case studies in the uniform PPR DSL, an engineering artifact from CPPS engineering that allows the representation of variants. Furthermore, we reported how the PPR DSL artifacts can be transformed to feature models automatically. We compared the case study feature models and the PPR DSL artifacts to investigate their complexity and expressiveness. We argue that the presented case studies with the PPR DSL instances and the flexible transformation capabilities build a valuable contribution to making future research results more comparable and facilitating evaluations with real-world product lines. We plan to extend our work incorporating other variability model approaches supported by TRAVART like decision modeling, e.g., for process variability modeling. We envision the community to use the case study materials as a starting point to evaluate their variability modeling approaches. Another use case of our real-world case studies is teaching product line engineering to students or training practitioners.

## ACKNOWLEDGMENTS

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. The competence center CDP is funded within the framework of COMET – Competence Centers for Excellent Technologies by BMVIT, BMDW, and the federal state of Vienna, managed by the FFG. The water filter was invented, designed, and produced by Askwar Hilonga at Gongalimodel in Tanzania. We thank Askwar Hilonga for providing and sharing this case. We gratefully acknowledge Yazgül Fidan's support, who conducted and analyzed the interviews with Askwar Hilonga. The truck models and their parts were designed, rendered, and 3D-printed at Czech Technical University in Prague - CIIRC. We thank Václav Jirkovský and Petr Novák for providing and sharing this case.

## REFERENCES

[1] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wąsowski. 2013. A survey of variability modeling in industrial practice. In *7th Int. Workshop on Variability Modelling of Software-intensive Systems*. ACM, 7–14.

[2] Thorsten Berger, Jan-Philipp Steghöfer, Tewfik Ziadi, Jacques Robin, and Jabier Martinez. 2020. The state of adoption and the challenges of systematic variability management in industry. *Empirical Software Engineering* 25 (2020), 1755–1797.

[3] Stefan Biffl, Detlef Gerhard, and Arndt Lüder. 2017. Introduction to the Multi-Disciplinary Engineering for Cyber-Physical Production Systems. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 1–24.

[4] Paul Clements and Linda Northrop. 2002. *Software product lines*. Addison-Wesley Boston.

[5] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. 2012. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. In *6th Int. Workshop on Variability Modeling of Software-Intensive Systems*. ACM, 173–182.

[6] Kevin Feichtinger, Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2020. Variability Transformation from Industrial Engineering Artifacts: An Example in the Cyber-Physical Production Systems Domain. In *24th ACM Int. Systems and Software Product Line Conf. - Volume B (SPLC '20)*. ACM, New York, NY, USA, 65–73.

[7] Kevin Feichtinger, Johann Stöbich, Dario Romano, and Rick Rabiser. 2021. TRAVART: An Approach for Transforming Variability Models. In *15th Int. Working Conf. on Variability Modelling of Software-Intensive Systems (VaMoS'21)*. ACM, New York, NY, USA, Article 8, 10 pages.

[8] Gongali Model. 2020. The Nanofilter ®. Gongali Model Co. Ltd Twitter. https://twitter.com/GongaliModel/status/1217727935744004096/photo/1.

[9] Lukas Kathrein, Arndt Lüder, Kristof Meixner, Dietmar Winkler, and Stefan Biffl. 2019. Production-Aware Analysis of Multi-disciplinary Systems Engineering Processes. In *21st Int. Conf. on Enterprise Information Systems - Vol. 2: ICEIS,*. INSTICC, SciTePress, 48–60.

[10] Beatrice Kiagho, Revocatus Machunda, Askwar Hilonga, and Karoli Njau. 2016. Performance of water filters towards the removal of selected pollutants in Arusha, Tanzania. *Tanzania Journal of Science* 42, 1 (2016), 134–147.

[11] Sebastian Krieter, Thomas Thüm, Sandro Schulze, Gunter Saake, and Thomas Leich. 2020. YASA: Yet Another Sampling Algorithm. In *14th Int. Working Conf. on Variability Modelling of Software-Intensive Systems*. ACM, 1–10.

[12] Jacob Krüger, Wolfram Fenske, Thomas Thüm, Dirk Aporius, Gunter Saake, and Thomas Leich. 2018. Apo-Games: A Case Study for Reverse Engineering Variability from Cloned Java Variants *(SPLC '18)*. ACM, New York, NY, USA.

[13] Jabier Martinez, Wesley K. G. Assunção, and Tewfik Ziadi. 2017. ESPLA: A Catalog of Extractive SPL Adoption Case Studies. In *21st Int. Systems and Software Product Line Conf. - Volume B (SPLC '17)*. ACM, New York, NY, USA, 38—-41.

[14] Jens Meinicke, Thomas Thüm, Reimar Schröter, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2017. *Mastering Software Variability with FeatureIDE*. Springer.

[15] Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2019. Towards Modeling Variability of Products, Processes and Resources in Cyber-Physical Production Systems Engineering. In *23rd Int. Systems and Software Product Line Conf. - Volume B*. ACM, 68:1–68:8.

[16] Kristof Meixner, Felix Rinker, Hannes Marcher, Jakob Decker, and Stefan Biffl. 2021. A Domain-Specific Language for Product-Process-Resource Modeling. In *IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*. IEEE.

[17] Marcilio Mendonca, Moises Branco, and Donald Cowan. 2009. SPLOT: software product lines online tools. In *24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*. 761–762.

[18] László Monostori. 2014. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP* 17 (2014), 9–13.

[19] Sana Ben Nasr, Guillaume Bécan, Mathieu Acher, João Bosco Ferreira Filho, Nicolas Sannier, Benoit Baudry, and Jean-Marc Davril. 2017. Automated extraction of product comparison matrices from informal product descriptions. *J. Syst. Softw.* 124 (2017), 82–103.

[20] Mikko Raatikainen, Juha Tiihonen, and Tomi Männistö. 2019. Software product lines and variability modeling: A tertiary study. *Journal of Systems and Software* 149 (2019), 485–510.

[21] Rick Rabiser and Alois Zoitl. 2021. Towards Mastering Variability in Software-Intensive Cyber-Physical Production Systems. *Procedia Computer Science, Proceedings of the 2nd Int. Conf. on Industry 4.0 and Smart Manufacturing (ISM 2020)* 180 (2021), 50–59.

[22] Per Runeson and Martin Höst. 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14 (2008), 131–164.

[23] Thomas Thüm and Fabian Benduhn. 2011. Spl2go: An online repository for open-source software product lines.

[24] Birgit Vogel-Heuser, Christoph Legat, Jens Folmer, and Stefan Feldmann. 2014. *Researching Evolution in Industrial Plant Automation: Scenarios and Documentation of the Pick and Place Unit*. Technical Report. Technische Universität München.

[25] Bernhard Wally, Jirí Vyskocil, Petr Novák, Christian Huemer, Radek Sindelár, Petr Kadera, Alexandra Mazak-Huemer, and Manuel Wimmer. 2021. Leveraging Iterative Plan Refinement for Reactive Smart Manufacturing Systems. *IEEE Trans Autom. Sci. Eng.* 18, 1 (2021), 230–243.

[26] Roel J. Wieringa. 2014. *Design science methodology for information systems and software engineering*. Springer.

### 5.3.3 Efficient Production Process Variability Exploration

#### Citation

[132] **K. Meixner**, K. Feichtinger, R. Rabiser, and S. Biffl. Efficient production process variability exploration. In F. A. Arcaini P., Devroey X., editor, *VaMoS '22: 16th International Working Conference on Variability Modelling of Software-Intensive Systems, Florence, Italy, February 23 - 25, 2022*, pages 14:1–14:9. Association for Computing Machinery, 2022. doi: 10.1145/3510466.3511274

#### Aim

The publication "Efficient Production Process Variability Exploration" introduces the IPSE approach for modeling and configuring models of variable product and production process assets. The publication provides (i) and elicitation of specific requirements for the integration of product, production process, and production resource variability integration, (ii) the IPSE approach to derive suitable state-of-the-art variability models for products and processes from a PPR–DSL model, and (iii) to efficiently explore the configuration space in a product feature model and a process decision model.

#### Contribution to the thesis

This publication contributes to the research goals CPPS knowledge model integration (**G1**.), PPR–DSL variability transformations (**G4**.), IRVM approach with CPPS variability exploration (**G3**., **G2**.), CPPS reuse method (**G2**.), and case study evaluation (**G5**.).

This publication contributes to **RQ2**., and **RQ3**. by addressing the VDI 3695 measures of M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *adaptability* C2., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of interdisciplinary collaboration* C5..

#### Abstract

CPPSs manufacture highly-customizable products from a product family following a sequence of production steps. For a CPPS, basic planners design feasible production process sequences by arranging atomic production steps based on implicit domain knowledge. However, the manual design of production sequences is inefficient and hard to reproduce due to the large configuration space. In this paper, we introduce the IPSE approach that (i) elicits domain knowledge in an industrial variability artifact, using the PPR–DSL; (ii) reduces configuration space size regarding structural product variability and behavioral process variability; and (iii) facilitates efficiently exploring the configuration space in a process decision model. For production process sequence design, IPSE is a first approach to combine structural and behavioral variability models. We investigated the feasibility of the IPSE in a study on a typical manufacturing work line in automotive production.

We compare the IPSE to a traditional process sequence planning approach. Our study indicates IPSE to be more efficient than the traditional manual approach.

# Efficient Production Process Variability Exploration

Kristof Meixner
CDL SQI, ISE
TU Wien
Austria
kristof.meixner@tuwien.ac.at

Kevin Feichtinger
LIT CPS Lab
JKU Linz
Austria
kevin.feichtinger@jku.at

Rick Rabiser
CDL VaSiCS, LIT CPS Lab
JKU Linz
Austria
rick.rabiser@jku.at

Stefan Biffl
Information Sys. Eng.
TU Wien and CDP
Austria
stefan.biffl@tuwien.ac.at

## ABSTRACT

Cyber-Physical Production Systems (CPPSs) manufacture highly-customizable products from a product family following a sequence of production steps. For a CPPS, basic planners design feasible production process sequences by arranging atomic production steps based on implicit domain knowledge. However, the manual design of production sequences is inefficient and hard to reproduce due to the large configuration space. In this paper, we introduce the *Iterative Process Sequence Exploration (IPSE) approach* that (i) elicits domain knowledge in an industrial variability artifact, using the Product-Process-Resource Domain-Specific Language (PPR–DSL); (ii) reduces configuration space size regarding structural product variability and behavioral process variability; and (iii) facilitates efficiently exploring the configuration space in a process decision model. For production process sequence design, IPSE is a first approach to combine structural and behavioral variability models. We investigated the feasibility of the IPSE in a study on a typical manufacturing work line in automotive production. We compare the IPSE to a traditional process sequence planning approach. Our study indicates IPSE to be more efficient than the traditional manual approach.

## CCS CONCEPTS

• **Software and its engineering → Software product lines**.

## KEYWORDS

Variability Modeling, Cyber-Physical Production System, Process Variability, Configuration Reduction.

**ACM Reference Format:**
Kristof Meixner, Kevin Feichtinger, Rick Rabiser, and Stefan Biffl. 2022. Efficient Production Process Variability Exploration. In *Proceedings of the 16th International Working Conference on Variability Modelling of Software-Intensive Systems (VAMOS '22), February 23–25, 2022, Florence, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3510466.3511274

## 1 INTRODUCTION

Software Product Line (SPL) engineering aims to systematically reuse and customize artifacts to support engineers at developing software-intensive systems [7]. Variability modeling aims to represent the commonalities, variability, and dependencies of reusable artifacts explicitly to create products sharing a common platform, i.e., a product line [3].

Cyber-Physical Production Systems (CPPSs), like automated car plants, use the latest information and communication technology and modern production techniques to manufacture highly-customizable products from a product line by following a sequence of production steps [30, 34]. *Basic planners*, key stakeholders in CPPS engineering, aim at designing feasible production process sequences to investigate their performance characteristics and estimate construction cost options for CPPS variants. The traditional approach is to manually design production process sequences according to implicit domain knowledge, e.g., for assembling car parts such as shift forks that require dozens of production steps. Unfortunately, this manual design process is hard to reproduce, especially with an evolving product line, e.g., with additional product variants, a key scenario in CPPS engineering [49]. Furthermore, this approach is inefficient due to the large space of possible but potentially undesirable or invalid process sequence solutions. Therefore, it is time-consuming and error-prone for domain experts to explore the (changed) configuration space without method and tool support.

A major challenge for automating parts of the production process design is the *insufficient representation of production process variability* due to missing dependencies to product variability [30]. Nevertheless, the variability and configuration of the products heavily determine the parameterization and/or sequence of the production steps. For instance, mounting a trunk lid to a car requires the chassis to be present. Depending on the car type, the trunk lid may be mounted at the top, bottom, or side, with different screw types, and at different production process stages. Therefore, it requires integrating heterogeneous (a) product variability, i.e., the CPPS can produce a variety of customized products, and (b) production process variability, i.e., the CPPS can produce products in different ways [4, 29]. However, current research on SPL engineering has been separately addressing either product variability [1, 38, 50] or process variability [44, 47].

Further, important domain-specific *process conditions,* such as vibration or temperature influences, *are hard to express formally*. This circumstance renders predefined [48] or fully automated [35] graph solutions infeasible in this context and requires domain experts to explore the solution space.

In this paper, we introduce the conceptual *Iterative Process Sequence Exploration (IPSE) approach* to integrate structural product variability and behavioral production process variability. IPSE aims at (a) eliciting domain knowledge in an industrial variability artifact to represent the variability of products and processes with their dependencies; (b) reducing the configuration space regarding structural product and behavioral process variability by leveraging explicit dependencies; and (c) facilitating the efficient and iterative exploration of the configuration space in a process decision model (DM).

We follow the Design Science methodology [52] to design the *IPSE* process, knowledge representation, and tool support. We evaluate the IPSE approach in a feasibility study on a typical manufacturing work line in automotive production with 19 production steps. To this end, we investigate the size of the process sequence configuration space and its reduction towards efficient iterative exploration by a domain expert. We measure the effort to conduct a task with a number of data elements, e.g., the effort to design a process sequence with 19 atomic process steps. We compare the study of the IPSE approach to a traditional manual process sequence planning approach to investigate strengths and limitations of IPSE.

The main contributions of this work are (a) an approach for configuring a process DM with a product feature model (FM) configuration (in a CPPS engineering context, i.e., production process sequence design); (b) novel TRAVART transformations from an industrial variability artifact to a DM; and (c) materials for reproducing IPSE from an abstracted representative use case in automotive manufacturing.[1]

The remainder of this work is structured as follows. Section 2 summarizes background and related work on CPPSs and on variability modeling. Section 3 presents an illustrative use case for evaluation. Section 4 presents the IPSE approach. Section 5.1 evaluates the IPSE approach. Section 5.2 discusses the results and implications considering related work. Section 6 concludes this work and raises topics for future work.

## 2 BACKGROUND AND RELATED WORK

This section provides a background and summarizes related work on CPPS engineering, variability modeling for CPPS, and variability model transformation.

### 2.1 CPPS Engineering

CPPSs are next-generation production systems that aim for the adaptive production of highly-customizable products [19, 30, 34]. CPPS engineering includes, beyond others, the life cycle phases of engineering the *product (line)* to be manufactured and the regarding *production system* [4]. Typically, the underlying product lines are subject to frequent evolution [4, 37], e.g., improving products or entering new markets with changed conditions. An essential task is production process engineering based on the product line [37].

Developing these software-intensive systems requires a collaboration of engineers from different domains, such as mechanical, electrical, and software engineering [4]. These experts create various artifacts, such as CAD drawings or bills of materials and operations, that model *Product-Process-Resource (PPR)* [45] aspects: (a) products with their parts, (b) production processes creating the products, and (c) production resources automating these processes. Some of these artifacts, such as type comparison matrices (TCMs), contain variant information on multiple levels, i.e., the product, production process, and production resource levels [13].

The Formal Process Description (FPD) [51] allows modeling PPR aspects as production process sequences with input and output products, their relation to production steps, and the required production resources. The relations also express process preconditions regarding previous product states and processes. Kathrein et al. [21] extended the FPD to model abstract aspects, dependencies and consistency constraints, such as the maximum force in a process. Meixner et al. [31] built on their work to design the Product-Process-Resource Domain-Specific Language (PPR–DSL) to facilitate the representation of PPR variants, i.e., variability information. This paper builds on the PPR–DSL to model products, their parts, and production process steps, including their functional process preconditions regarding previous processes.

In the first CPPS engineering phase, *basic planners* model a rough CPPS design including a feasible production process sequence according to several (often implicit) constraints. For example, it is prudent to mount costly product parts at a late production step where the scrap rate is lower than at an early production step. Therefore, the adaptive production of product lines requires the management of variability during CPPS engineering and at run time. The variability in CPPSs stems from (a) *product types* and their features, (b) *process (sequence) variants*, and (c) *resource variants*. Thus, the variability of the PPR aspects, their dependencies, and often non-functional constraints and feature cardinalities typically create a large and complex problem and configuration space [9, 39, 46]. For instance, a car consists of hundreds of thousands of components assembled in several thousands of production steps. We focus on variability resolved during CPPS engineering before the CPPS goes into operation.

### 2.2 Variability Modeling for CPPS

The most prominent Variability Modeling (VM) approach is, de facto, feature modeling [20]. Decision modeling [25] has also been widely investigated [8]. Both approaches have been refined and extended in the last decades [8, 40].

There is a plethora of Systematic Literature Reviews (SLRs), mapping studies, and surveys that investigate a broad range of VM methods and models [2, 3, 6, 8, 40]. Raatikainen et al. [40] and Galster et al. [16] report that process model variability and business processes as artifacts are underrepresented. Rosa et al. [43] conducted a survey on VM for business processes, showing that only a few business process model types have been investigated. Early, Rombach [42] already proposed integrating software process and product lines as a future vision, which has, however, not yet been achieved.

Several works [10, 18, 24, 32, 33] report on staged configuration and feature configuration flows that utilize FMs for an iterative configuration. These and others [17] also stress that the separation of concerns is important, especially, in multidisciplinary VM, required for CPPS engineering.

A key challenge is allowing *basic planners* to use the PPR–DSL model without forcing them to use a new or unfamiliar approach while linking structural and behavioral variability using well-known models. Existing VM approaches mainly focus either on structural

[1]The supplementary material to this work can be found online at: https://github.com/tuw-qse/cpps-variability

OR behavioral variability and are *as-is* incapable to handle CPPS engineering variability and the combination of different CPPS aspects [8]. In this paper, we generate (a) a FM for the structural product variability and (b) a DOPLER DM for the behavioral process variability from the PPR–DSL artifacts and (c) automatically link both models. This way, we aim at investigating staged configuration to a certain point, using FM and DM to suit these demands.

FMs [8] are well suited to represent structural variability since hierarchy is a key concept in these approaches. However, they have not been designed to define behavioral variability. Dhungana et al. [11] presented the DOPLER approach, a flexible and extensible DM-based approach using the DOPLER VM language. The DOPLER DM describes the decisions that need to be made during product derivation to configure assets, such as code artifacts. The approach allows defining orders of making decisions via *visibility conditions* (decisions become "visible", i.e., relevant, depending on other decisions' values) and decision rules (making decisions affect other decisions' values and value ranges).

### 2.3 Variability Model Transformation

Generating variability models requires a model transformation approach for the PPR–DSL models. Building on *variability meta-models* [5, 20], the TRAVART [15] approach specifies *transformation operations* implemented in concrete *transformation algorithms* to transform one (type of) model into another one [14]. In absence of a meta-model, TRAVART allows specifying concrete *transformation operations*. For instance, for custom variability representations, like Domain-Specific Languages (DSLs) artifacts, product comparison matrices (PCMs) [36], or TCMs, such *transformation operations* can be specified, to transform them into, e.g., a FM.

In this work, we employ the TRAVART approach [15]. We build on the *transformation operations* specified for transforming the product variants in the PPR–DSL into a FM [28] and specify similar operations for transforming production process steps in the PPR–DSL into a DOPLER DM.

## 3 ILLUSTRATIVE USE CASE FOR EVALUATION

To illustrate and evaluate process sequence exploration for a comprehensive, concise, and still manageable example, we adapted our previous *shift fork* case study [28]. In the manual transmission of a car, shift forks shift cuffs along pipes to correctly align the gears.[2] *Shift forks* establish a product line with variable product parts, such as pipes of different lengths. We adapted the study with domain experts from our CPPS industry partner to check the correctness and get feedback.

In CPPS engineering, *basic planners* receive design artifacts, such as CAD drawings or prototypes of products, from their customers. They explore the product similarities and variability, including dependencies between the product types and their parts. TCMs are intermediate design artifacts that contain product types and their required parts. From a TCM, basic planners can derive similar functions of the product parts, e.g., parts that resemble a sub-type of a pipe.

[2] An exemplary figure of *shift forks* can be found at https://w.wiki/3DCf

The engineers also investigate the production steps required for the particular product types, often by first disassembling them [23]. They aim at designing first drafts of feasible production processes for requested products with corresponding CPPS layouts and a rough cost estimate. Based on the dependencies between the product parts, a basic planner models a precedence graph of how product types imply a particular assembly order, e.g., parts that need to be mounted onto a pipe. From this graph, the planner derives atomic production steps that transform input products into output products, e.g., joining a fork to a pipe. In the use case context, traditionally, a basic planner designs feasible process sequences, e.g., in DelMia[3], based on implicit knowledge, typically using heterogeneous and partial data representations. The planner uses the sequences to reason on process characteristics, e.g., duration or cost of faulty products.

```
Product "Pipe": { name: "Abstract Pipe", isAbstract: true }
Product "Pipe2": { name: "Pipe 2",
    implements: ["Pipe"],
    excludes: [ "Pipe3", "Pipe8" ]
}

Product "Lock": { name: "Abstract Lock",
    isAbstract: true , requires: ["Pipe"]
}
Product "Lock1": { name: "Lock 1",
    implements: ["Lock"],
    excludes: [ "Lock2", "Lock3" ]
}

Process "InsertPipe2": { name: "InsertPipe2",
    isAbstract: false ,
    implements: ["InsertPipe"],
    inputs: [ {productId: "Pipe2"} ],
    outputs: [ {OP2: {productId: "Pipe2"}} ]
}

Process "WeldLock": { name: "WeldLock",
    isAbstract: true ,
    requires: [ "InsertLock", "InsertPipe", ... ],
    inputs: [ {productId: "Lock"}, {productId: "Pipe"} ],
    outputs: [ {OP22: {productId: "ForkProduct"}}]
}
```

**Listing 1: PPR DSL [31] excerpt for the *shift fork* use case.**

However, due to frequent *product line evolution*, engineers need to investigate the impact of the changes on the production sequence and the CPPS design [37]. Often a basic planner, different from the initial planner, conducts this process redesign. According to practitioners, this often results in rebuilding the CPPS design from scratch rather than reusing existing solutions as the engineering process is hard to reproduce. Hence, this approach is inefficient and prone to error due to the limited representation of domain knowledge that hinders systematic improvement, automation, and reuse.

To explicitly model their domain knowledge, *basic planners* can use the PPR–DSL [31] to model the PPR aspects and constraints between them. The use case consists of four shift fork types with 14 product parts. Furthermore, the product parts each have up to three dependencies to other parts.

[3] https://www.3ds.com/products-services/delmia/

Listing 1 illustrates the PPR–DSL model of selected *shift fork* parts: product parts and variants (lines 1-13); and atomic production steps and variants (lines 15-27). Line 1 defines the abstract Product *Pipe*, building the core of a *shift fork*. Product *Pipe2* implements the abstract *Pipe* and excludes other product parts of type pipe, defining an alternative group. Lines 7 to 9 define the abstract *Lock* and concrete *Lock1* part. Lines 10 to 13 define the *Lock* to require a *Pipe* for assembly. Lines 15 to 20 model the Process *InsertPipe2*, that implements *InsertPipe*, with its input and output products, i.e., *Pipe2*. *InsertPipe* changes the location of *Pipe2* in the system. Lines 22 to 27 define the abstract process *WeldLock* with *Lock* and *Pipe* as inputs. *WeldLock* requires *InsertLock* and *InsertPipe* as preconditions.

For this paper, we assume *basic planners* able to describe the atomic process steps and their preconditions for all product types and parts.

## 4 SOLUTION: THE IPSE APPROACH

This section identifies requirements for process sequence exploration for product lines and introduces the five-step *Iterative Process Sequence Exploration (IPSE)*. We elicited the requirements and designed the IPSE process based on the literature and a domain analysis with experts from an industry partner. In the domain analysis, we interviewed three senior domain experts on their traditional manual engineering approach, which artifacts they produce, and their issues with process sequence exploration in CPPS engineering.

### 4.1 Requirements for IPSE

From the literature [13, 30] and the domain analysis, we elicited the following requirements for an IPSE process with data representation and tool support. *R1. The IPSE process* shall collect and transform the variability knowledge required for efficiently exploring production process sequence options. In particular, the IPSE shall consider new knowledge iteratively, especially from product line evolution but also, e.g., process simulation. *R2. The IPSE knowledge representation* shall model (a) products, atomic process steps, and their dependencies in an industrial variability artifact, e.g., using the PPR–DSL [31]; (b) a product line; and (c) a process line with process sequence dependencies and constraints. *R3. The IPSE method and tool support* shall (a) automate transformations from industrial variability artifacts to well-known variability models, e.g., a FM or a DM, using frameworks, such as TRAVART [15]; (b) provide assistance to explore and configure a product FM and a process DM, such as the FeatureIDE [26] and DOPLER tool [11].

### 4.2 The IPSE Process

Based on the literature [13] and the domain analysis, we developed the IPSE process (cf. Figure 1). IPSE comprises steps conducted by engineers with tool support (persona icon) and automated steps (cog icon). IPSE, addressing requirement *R1*, provides tool support in form of an *IPSE agent*. That agent conducts some tasks automatically and supports the basic planner in the "human" tasks that require access to models. We describe this process in detail in the following.

**Step 1: Define PPR–DSL model.** The *basic planner* analyzes product descriptions and production requirements provided by the product designer and customer (cf. Section 3) to identify atomic production steps required to assemble a (partial) product variant. The *basic planner* iteratively models in the PPR–DSL model (a) the product variants, their parts, (b) the atomic process steps with their input and output products, as required to assemble the products, and (c) dependencies between a process step and preceding abstract/concrete products and processes (cf. Listing 1) – (requirement *R2*).

For instance, the basic planner defines the fork as laser-welded to a pipe by a welding robot to achieve a reliable connection for the shift fork. Preconditions for the process step laser-welding are the fork and the pipe, correctly inserted into the CPPS (cf. Listing 1, line 25). In companies with advanced reuse capabilities, the *basic planner* can select and adapt atomic process steps from a process catalog.

**Step 2: Transform PPR–DSL model to FM and DM.** In this step, the IPSE agent utilizes the TRAVART approach [15] to transform the PPR–DSL model into two different variability models (cf. requirement *R2* & *R3* in Section 4.1): (a) a FM to represent the product parts and variants (cf. Figure 2); and (b) a DM to represent the atomic process steps and their dependencies (cf. Table 1).

**Step 2a: *PPR–DSL to* Product *FM*.** Input to this step is the PPR–DSL model containing the products, their parts, and the atomic production process steps. The result of this step is a *Product* FM that contains the products and their parts as features, including constraints between the product parts.

The *IPSE agent* utilizes TRAVART transformations [15] to generate a FM from the product definitions. The transformations create a feature for each product part and derive its properties, e.g., whether the feature is abstract or mandatory, from the respective PPR–DSL properties, e.g., isAbstract (cf. Listing 1, line 1). For instance, the product *Pipe* is transformed to an abstract feature with a concrete child feature *Pipe2*. Based on the properties requires and excludes, the transformation derives the feature group cardinality and/or additional constraints. For example, the features *Pipe2* to *Pipe8* build an alternative group. Afterwards, a transformation generates the relevant constraints in the FM, e.g., a particular *Lock* implies the respective feature *Pipe*, e.g., *Lock2* implies *Pipe3*.

**Step 2b: *PPR–DSL to* Process *DM*.** Input to this step is the PPR–DSL model containing the products, their parts, and the atomic production process steps with their dependencies. The result of this step is a *Process* DM with decisions concerning the products and process steps for possible production sequences for a product line. The IPSE agent utilizes TRAVART transformations [15] to transform the PPR–DSL model into the DM in two transformation stages.

The *first transformation stage* translates the abstract and concrete products of the PPR–DSL model into Boolean product-related decisions in the DM (cf. Table 1, upper half). The transformation assigns these decisions with a Boolean visibility condition, initially set to false. This means, while a particular decision is not visible in the configuration view, its configuration value can be used in the DM. For instance, the product parts *Pipe2* and *Lock1* are transformed to such invisible decisions in the DM. The configuration value of the particular decision, i.e., whether the decision is selected, is then set in *Step 4* of the IPSE process.

The *second transformation stage* generates Boolean process decisions (cf. Table 1, lower half) from the abstract and concrete PPR–DSL process steps. The transformation sets the visibility condition

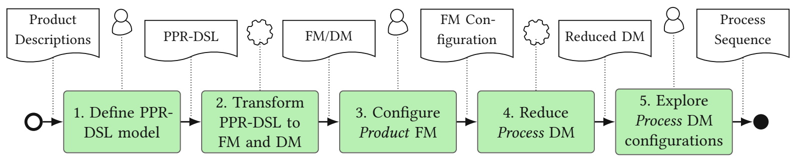


**Figure 1: (Human & automated) IPSE process steps for exploring production process steps based on a product configuration.**

of each abstract process decision to false, ensuring only concrete decisions to be visible in the configuration view. The visibility condition of each concrete process decision is calculated from the configuration values for the required input products, i.e., inputs, and the required production process steps in preconditions, i.e., requires. The visibility condition results from the logical and concatenation of the involved Boolean configuration values. For instance, the transformation of the process step *WeldLock* shall result in a decision *Do you want to WeldLock?* The visibility condition of this decision shall concern the products *Lock* and *Pipe* and the process steps *InsertLock* and *InsertPipe*. The transformation also creates constraints for concrete process steps in the DM that refer to the abstract process steps. For instance, selecting the decision *InsertPipe2* automatically sets the configuration value for the abstract decision *InsertPipe*. These constraints ensure considering abstract decisions during the configuration, such as references to abstract products.

**Step 3: Configure *Product* FM.** In this step, the *basic planner* configures the *Product* FM by selecting the desired features. The result is a valid configuration of the *Product* FM that represents one particular (final target) product of the product line, e.g., a specific type of shift fork.

**Step 4: Reduce *Process* DM.** Based on the configuration of the *Product* FM, the *IPSE agent* reduces (cf. requirement *R3*) the configuration possibilities of the DM to the valid subset of process sequence configurations for a particular product configuration. The result is the reduced *Process* DM.

For each feature in the *Product* FM configuration, the configuration value of the particular product decision in the *Process* DM is set to true. For example, assume in the *Product* FM the product *Pipe2* feature to be selected. The *IPSE agent* then sets the configuration value of the *Pipe2* product decision in the DM to true. This change triggers displaying decisions with visibility conditions that depend on that particular configuration value, e.g., process decision *InsertPipe2*.

**Step 5: Explore *Process* DM configurations.** Input to this step is the reduced *Process* DM, configured to a target product. The *basic planner* can iteratively explore the configuration options for the complete production process sequence. The result is one valid production process sequence to produce the target product.

Initially, an IPSE tool (requirement *R3*) facilitates browsing the visible potential first process steps, i.e., steps without preconditions, for the product configured in the FM. After configuring a first process step decision, i.e., the first process step to execute, the basic planner sees the potential subsequent process steps, according to the updated visibility conditions defined by the process preconditions. The *basic planner* can then explore the DM and decide which process steps to execute one after another. In the end, all necessary decisions to manufacture the product configured in the *Product* FM will have been taken in a sequence by the basic planner. The IPSE agent keeps a queue with the sequence of decisions taken, which can be shown as an intermediate result towards an increasingly complete and valid assembly sequence. The final sequence of the configuration decisions reflects the sequence of the desired production steps, which can be connected to define a valid production process model.

## 5 EVALUATION AND DISCUSSION

This section reports on a feasibility study on IPSE and discusses research results and limitations.

### 5.1 Feasibility study

The authors conducted the IPSE process steps in a feasibility study[1] regarding the *shift fork* case study selected from an existing repository with CPPS case studies [28]. Further, the authors discussed the study results with senior domain experts from an industry partner in the CPPS domain and an expert in systems engineering for initial validation.

**Step 1. Define PPR–DSL model.** One author extended the PPR–DSL artifact of the *shift fork* case study (cf. Section 3) with atomic process steps [12] and their dependencies. The resulting PPR–DSL model contains 23 product and part definitions, 28 atomic process steps, 3 constraints, and 86 dependencies from processes to products or processes. It took the author around 8 to 10 work hours to define the PPR–DSL model for the use case based on existing information in documents. To ensure that the particular process steps are suitable and the overall process is feasible, we interviewed the domain experts to approve the resulting model.

**Step 2a. Transform PPR–DSL to *Product* FM.** One author used the existing TRAVART transformations and tool to transform the product parts, described in the PPR–DSL, into a FeatureIDE FM [13], requiring roughly one work hour for preparation and execution. The transformation consisted of less than 400 transformation steps that took seconds to execute on a standard notebook. Figure 2 shows for the *shift fork* the generated *Product* FM with its constraints.[1]

**Step 2b. Transform PPR–DSL to *Process* DM.** One author implemented the necessary TRAVART transformation operations towards a DOPLER DM [11] based on existing operations in around 8 to 10 work hours. For future transformation to a *Process* DM, these

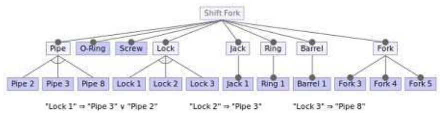


**Figure 2: *Product* FM, generated from the PPR–DSL model, representing the structural *shift fork* type variability.**

transformation operations can be reused. TRAVART transformed the *shift fork* PPR–DSL model to a DOPLER *Process* DM. The transformation consisted of around 250 transformation steps that took seconds to execute on a standard notebook.

Table 1 illustrates an excerpt of the resulting *Process* DM (cf. supplemental material for the complete DM[1]). Column *ID* shows the decision's ID. Column *Question* shows the question generated for display to users of the DM. Column *Decision Rule* shows a distinguished feature of the DOPLER approach [11], the DOPLER decision rule, which is executed when the decision is taken. For instance, for the *InsertPipe2* decision a rule sets the configuration value of the decision for the abstract process step *InsertPipe* to true. This ensures to set decisions to visible with visibility conditions that depend on abstract products or process steps. Column *Visible/relevant if* shows the logical expression for the visibility condition. For example, the decision *InsertPipe2* is only displayed if the configuration value with the ID *Pipe2* is set to true.

**Step 3. Configure *Product* FM.** For the feasibility study, one author sampled with the YASA sampler [22] all valid configurations of the *shift fork* FM, resulting in four product configurations, as an input for *Step 4*. This way, the authors ensure finding any desired number of configurations for further evaluations. While the *shift fork* had only four possible configurations, the authors' aim was to use this strategy to sample larger models in the future. The effort for the sampler implementation was roughly 35 to 40 work hours; product sampling took less than 10 seconds. For the feasibility study, the authors conducted Steps 4 and 5 for all four sampled product configurations.

**Step 4. Reduce *Process* DM.** For this step of the study, one author implemented a Java program to reduce the *Process* DM, requiring around 8 to 10 work hours. The program gets a configuration of a FeatureIDE FM and sets the corresponding configuration values in a DOPLER DM to true. In the feasibility study, these were configuration values for the product decisions. The result for the feasibility study was a *Process* DM for each product configuration.

**Step 5. Explore *Process* DM configurations.** One author implemented a DOPLER DM sampler, which took around three work days, roughly 20 to 25 work hours. to sample valid configurations of a *Process* DM. This sampler receives a DM and takes the decisions using backtracking to compute all possible configurations of the *Process* DM.

For the feasibility study, one author sampled the transformed *Process* DM. For each FM configuration, the sampler found around 59k possible process sequences, i.e., total configurations of the DM. Only 1,024 (under 2%) were valid process sequences yielding a valid final product. The four product configurations resulted in the same number of possible and valid process sequences. This outcome is reasonable for the studied product line, as for each final product two similar parts were changed (the pipe and the lock).

One author, who did not define the PPR–DSL model, explored a complete production process sequence with 19 decisions with 7 initial, on average around 3, decision options. We measured the time required for investigating a production process step and its possible successors. Selecting one production step took only 30 seconds to one minute as the *Process* DM provided only valid process steps options for selection. The overall time required for configuring for one complete production process sequence was 15 to 20 minutes. However, this was without considering particular engineering characteristics of the process variants, i.e., hard-to-represent domain expert knowledge, such as vibration dependencies.

**Traditional vs IPSE approach.** Traditionally, the domain experts create the product variants and atomic production process steps, e.g., using tools such as DelMia[3]. This step is comparable to the PPR–DSL model definition in IPSE **Step 1**. While DelMia allows modeling *parent-child* relationships, the PPR–DSL also allows to define extended constraints, e.g., requires, and constraints on properties. Depending on the extensiveness of the constraints that engineers want to express, this effort needs to be added for IPSE Step 1. The IPSE allows starting with a small PPR–DSL model (comparable to the DelMia model) that can be iteratively extended regarding the level of detail on dependencies and constraints. Here, domain experts need to balance the effort to model domain knowledge for solution space reduction vs. the effort to explore a larger space of infeasible solutions.

Traditionally, the engineers define a valid production process sequence by connecting the atomic process steps, e.g., in DelMia[3], using their implicit knowledge. However, they can connect incompatible process steps, requiring the engineer to ensure the compatibility of steps. IPSE uses the order of the process steps given by the explicit constraints to allow connecting compatible process steps only.

**Changes to the product line** need to be incorporated in the DelMia model and the PPR–DSL model, with comparable effort. Note that the effort partly depends on the complexity of the constraints expressed in the PPR–DSL model.

For the production process sequence exploration (IPSE **Step 5**) in the traditional approach, the production process sequence for a particular product type needs to be re-defined, according to the domain experts often from scratch, due to the missing explicit knowledge. Here, the product evolution and the impact on the processes are drivers of the process design effort. In the IPSE approach, the explicit models lead to exploring only valid production process sequences, making the exploration process less error-prone and making the decisions easier to reproduce.

### 5.2 Discussion

This work investigated the IPSE approach to support CPPS engineering in efficiently configuring valid production process sequences, given an evolving product line. Therefore, we introduced the technical concept of the IPSE approach in Section 4.2 addressing the requirements from Section 4.1.

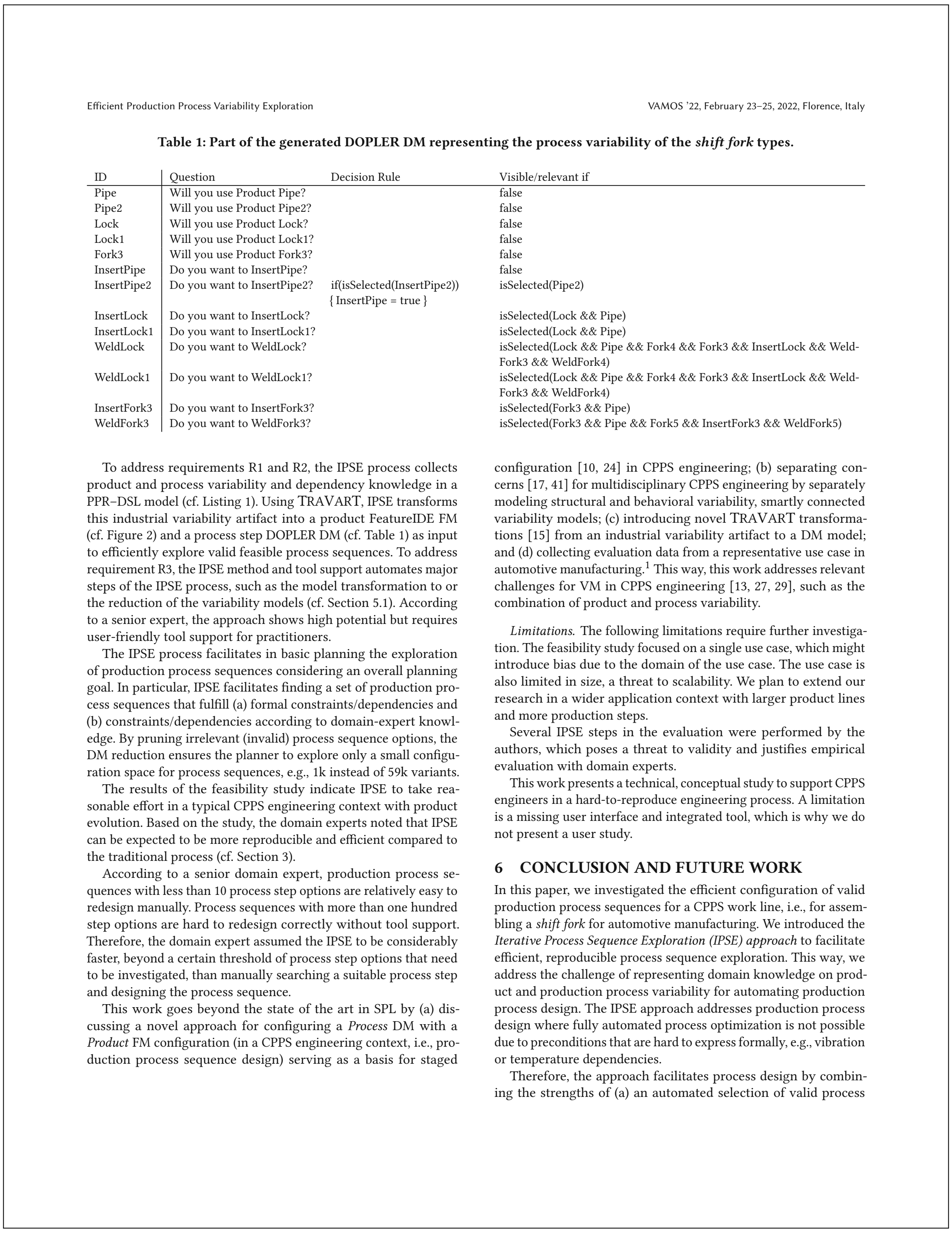


**Table 1: Part of the generated DOPLER DM representing the process variability of the *shift fork* types.**

| ID | Question | Decision Rule | Visible/relevant if |
|---|---|---|---|
| Pipe | Will you use Product Pipe? | | false |
| Pipe2 | Will you use Product Pipe2? | | false |
| Lock | Will you use Product Lock? | | false |
| Lock1 | Will you use Product Lock1? | | false |
| Fork3 | Will you use Product Fork3? | | false |
| InsertPipe | Do you want to InsertPipe? | | false |
| InsertPipe2 | Do you want to InsertPipe2? | if(isSelected(InsertPipe2)) { InsertPipe = true } | isSelected(Pipe2) |
| InsertLock | Do you want to InsertLock? | | isSelected(Lock && Pipe) |
| InsertLock1 | Do you want to InsertLock1? | | isSelected(Lock && Pipe) |
| WeldLock | Do you want to WeldLock? | | isSelected(Lock && Pipe && Fork4 && Fork3 && InsertLock && WeldFork3 && WeldFork4) |
| WeldLock1 | Do you want to WeldLock1? | | isSelected(Lock && Pipe && Fork4 && Fork3 && InsertLock && WeldFork3 && WeldFork4) |
| InsertFork3 | Do you want to InsertFork3? | | isSelected(Fork3 && Pipe) |
| WeldFork3 | Do you want to WeldFork3? | | isSelected(Fork3 && Pipe && Fork5 && InsertFork3 && WeldFork5) |

To address requirements R1 and R2, the IPSE process collects product and process variability and dependency knowledge in a PPR–DSL model (cf. Listing 1). Using TRAVART, IPSE transforms this industrial variability artifact into a product FeatureIDE FM (cf. Figure 2) and a process step DOPLER DM (cf. Table 1) as input to efficiently explore valid feasible process sequences. To address requirement R3, the IPSE method and tool support automates major steps of the IPSE process, such as the model transformation to or the reduction of the variability models (cf. Section 5.1). According to a senior expert, the approach shows high potential but requires user-friendly tool support for practitioners.

The IPSE process facilitates in basic planning the exploration of production process sequences considering an overall planning goal. In particular, IPSE facilitates finding a set of production process sequences that fulfill (a) formal constraints/dependencies and (b) constraints/dependencies according to domain-expert knowledge. By pruning irrelevant (invalid) process sequence options, the DM reduction ensures the planner to explore only a small configuration space for process sequences, e.g., 1k instead of 59k variants.

The results of the feasibility study indicate IPSE to take reasonable effort in a typical CPPS engineering context with product evolution. Based on the study, the domain experts noted that IPSE can be expected to be more reproducible and efficient compared to the traditional process (cf. Section 3).

According to a senior domain expert, production process sequences with less than 10 process step options are relatively easy to redesign manually. Process sequences with more than one hundred step options are hard to redesign correctly without tool support. Therefore, the domain expert assumed the IPSE to be considerably faster, beyond a certain threshold of process step options that need to be investigated, than manually searching a suitable process step and designing the process sequence.

This work goes beyond the state of the art in SPL by (a) discussing a novel approach for configuring a *Process* DM with a *Product* FM configuration (in a CPPS engineering context, i.e., production process sequence design) serving as a basis for staged configuration [10, 24] in CPPS engineering; (b) separating concerns [17, 41] for multidisciplinary CPPS engineering by separately modeling structural and behavioral variability, smartly connected variability models; (c) introducing novel TRAVART transformations [15] from an industrial variability artifact to a DM model; and (d) collecting evaluation data from a representative use case in automotive manufacturing.[1] This way, this work addresses relevant challenges for VM in CPPS engineering [13, 27, 29], such as the combination of product and process variability.

*Limitations.* The following limitations require further investigation. The feasibility study focused on a single use case, which might introduce bias due to the domain of the use case. The use case is also limited in size, a threat to scalability. We plan to extend our research in a wider application context with larger product lines and more production steps.

Several IPSE steps in the evaluation were performed by the authors, which poses a threat to validity and justifies empirical evaluation with domain experts.

This work presents a technical, conceptual study to support CPPS engineers in a hard-to-reproduce engineering process. A limitation is a missing user interface and integrated tool, which is why we do not present a user study.

## 6 CONCLUSION AND FUTURE WORK

In this paper, we investigated the efficient configuration of valid production process sequences for a CPPS work line, i.e., for assembling a *shift fork* for automotive manufacturing. We introduced the *Iterative Process Sequence Exploration (IPSE) approach* to facilitate efficient, reproducible process sequence exploration. This way, we address the challenge of representing domain knowledge on product and production process variability for automating production process design. The IPSE approach addresses production process design where fully automated process optimization is not possible due to preconditions that are hard to express formally, e.g., vibration or temperature dependencies.

Therefore, the approach facilitates process design by combining the strengths of (a) an automated selection of valid process sequences from a large solution space based on a product configuration with (b) the skills of a domain expert to select promising candidates from valid process steps using their advanced domain knowledge.

Major contributions of this work are that IPSE (a) elicits domain knowledge in an industrial variability artifact, using the PPR–DSL; (b) transforms the variability to well-known variability models, the *Product* FM and *Process* DM; (c) significantly reduces the configuration space size regarding structural product variability and behavioral process variability; and (d) facilitates efficiently exploring the configuration space in the *Process* DM. To this end, the IPSE goes beyond the state of the art in SPL engineering for CPPS by integrating product and process variability for production process design and fostering a separation of concerns [13, 27, 29].

Results from a feasibility study on a work line in automotive production with 19 production steps, based on an industrial case from our industry partner [28], indicate IPSE to be (a) feasible; (b) better reproducible in contrast to the traditional manual process sequence exploration (cf. Section 5.1); and (c) more efficient by finding the 2% of valid configurations in the *shift fork* use case for iterative exploration. According to a senior domain expert, a similar reduction can be expected for larger work lines.

**Future Work.** The promising results of the initial evaluation warrant conducting case studies in a wider context to validate the initial evaluation results. We plan to investigate the IPSE approach with further domain experts and typical instances of large work lines with 50 to 100 process steps and, thus, much larger solution spaces than with the use case *shift fork*. To further improve the efficiency of the IPSE, we plan to investigate advanced capabilities for process sequence exploration. Such capabilities are, e.g., backtracking, managing promising (partial) result sets, and evaluating process characteristics, e.g., process optimization criteria, such as throughput, duration, or cost, and considering process simulation results or heuristics in the background.

## ACKNOWLEDGMENTS

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) via "Austrian Competence Center for Digital Production" (CDP) under contract nr. 881843.

## REFERENCES

[1] Sven Apel, Don Batory, Christian Kaestner, and Gunter Saake. 2013. *Feature-Oriented Software Development: Concepts and Implementation.* Springer.

[2] Rabih Bashroush, Muhammad Garba, Rick Rabiser, Iris Groher, and Goetz Botterweck. 2017. CASE Tool Support for Variability Management in Software Product Lines. *Comput. Surveys* 50, 1 (March 2017), 14:1–14:45.

[3] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wąsowski. 2013. A survey of variability modeling in industrial practice. In *7th International Workshop on Variability Modelling of Software-intensive Systems.* ACM, 7–14.

[4] Stefan Biffl, Detlef Gerhard, and Arndt Lüder. 2017. Introduction to the Multi-Disciplinary Engineering for Cyber-Physical Production Systems. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems.* Springer, 1–24.

[5] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. 2017. Model-driven software engineering in practice. *Synthesis Lectures on Software Engineering* 3, 1 (2017), 1–207.

[6] Lianping Chen and Muhammad Ali Babar. 2011. A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology* 53, 4 (April 2011), 344–362.

[7] Paul Clements and Linda Northrop. 2002. *Software product lines.* Addison-Wesley Boston.

[8] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. 2012. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. In *6th International Workshop on Variability Modeling of Software-Intensive Systems.* ACM, 173–182.

[9] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. 2005. Formalizing cardinality-based feature models and their specialization. *Software process: Improvement and practice* 10, 1 (2005), 7–29.

[10] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. 2005. Staged configuration through specialization and multilevel configuration of feature models. *Software process: improvement and practice* 10, 2 (2005), 143–169.

[11] Deepak Dhungana, Paul Grünbacher, and Rick Rabiser. 2011. The DOPLER Meta-Tool for Decision-Oriented Variability Modeling: A Multiple Case Study. *Automated Software Engineering* 18, 1 (2011), 77–114.

[12] DIN. 2003. DIN 8580 – Manufacturing processes. https://standards.globalspec.com/std/756719/DIN%208580 [Online; accessed 2019-04-02].

[13] Kevin Feichtinger, Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2020. Variability Transformation from Industrial Engineering Artifacts: An Example in the Cyber-Physical Production Systems Domain. In *3rd International Workshop on Variability and Evolution of Software-Intensive Systems (VariVolution), co-located with SPLC 2020.* ACM, 65–73.

[14] Kevin Feichtinger and Rick Rabiser. 2020. Towards Transforming Variability Models: Usage Scenarios, Required Capabilities and Challenges. In *24th ACM International Systems and Software Product Line Conference - Volume B (SPLC '20).* ACM, 44–51.

[15] Kevin Feichtinger, Johann Stöbich, Dario Romano, and Rick Rabiser. 2021. TRAVART: An Approach for Transforming Variability Models. In *15th International Working Conference on Variability Modelling of Software-Intensive Systems.* ACM, 8:1–8:10.

[16] Matthias Galster, Danny Weyns, Dan Tofan, Bartosz Michalik, and Paris Avgeriou. 2014. Variability in Software Systems - A Systematic Literature Review. *IEEE Transactions on Software Engineering* 40, 3 (March 2014), 282–306.

[17] Gerald Holl, Paul Grünbacher, and Rick Rabiser. 2012. A Systematic Review and an Expert Survey on Capabilities Supporting Multi Product Lines. *Information and Software Technology* 54, 8 (2012), 828–852.

[18] Arnaud Hubaux, Andreas Classen, and Patrick Heymans. 2009. Formal modelling of feature configuration workflows. In *SPLC (ACM International Conference Proceeding Series, Vol. 446).* ACM, 221–230.

[19] Didac Gil De La Iglesia and Danny Weyns. 2015. MAPE-K Formal Templates to Rigorously Design Behaviors for Self-Adaptive Systems. *ACM Transactions on Autonomous and Adaptive Systems* 10, 3 (Sept. 2015).

[20] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson. 1990. *Feature-oriented domain analysis (FODA) feasibility study.* Technical Report. Carnegie-Mellon Univ., Pittsburgh, Pa, Software Engineering Inst.

[21] Lukas Kathrein, Arndt Lüder, Kristof Meixner, Dietmar Winkler, and Stefan Biffl. 2019. Production-Aware Analysis of Multi-disciplinary Systems Engineering Processes. In *21st International Conference on Enterprise Information Systems - Vol. 2: ICEIS,.* INSTICC, SciTePress, 48–60.

[22] Sebastian Krieter, Thomas Thüm, Sandro Schulze, Gunter Saake, and Thomas Leich. 2020. YASA: Yet Another Sampling Algorithm. In *14th International Working Conference on Variability Modelling of Software-Intensive Systems.* ACM, 1–10.

[23] Sukhan Lee. 1989. Disassembly planning based on subassembly extraction. In *Third ORSA/TIMS Conference on Flexible Manufacturing System.* 383–388.

[24] Malte Lochau, Johannes Bürdek, Stefan Hölzle, and Andy Schürr. 2017. Specification and automated validation of staged reconfiguration processes for dynamic software product lines. *Software and Systems Modeling* 16, 1 (2017), 125–152.

[25] Rich McCabe, Grady Campbell, Neil Burkhard, Steve Wartik, Jim O'Connor, Joe Valent, and Jeff Facemire. 1993. *Reuse-Driven Software Processes Guidebook.* techreport SPC-92019-CMC, Version 02.00.03. Software Productivity Consortium.

[26] Jens Meinicke, Thomas Thüm, Reimar Schröter, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2017. *Mastering Software Variability with FeatureIDE.* Springer.

[27] Kristof Meixner. 2020. Integrating Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering. In *24th ACM International Systems and Software Product Line Conference - Volume B (SPLC '20).* ACM, 96–103.

[28] Kristof Meixner, Kevin Feichtinger, Rick Rabiser, and Stefan Biffl. 2021. A Reusable Set of Real-World Product Line Case Studies for Comparing Variability Models in Research and Practice. In *25th ACM International Systems and Software Product Line Conference - Volume B (SPLC '21).* ACM, 105–112.

[29] Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2019. Towards modeling variability of products, processes and resources in cyber-physical production systems engineering. In *23rd International Systems and Software Product Line Conference - Volume B (SPLC '19).* ACM, 68:1–68:8.

[30] Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2020. Feature Identification for Engineering Model Variants in Cyber-Physical Production Systems Engineering.

In *14th International Working Conference on Variability Modelling of Software-Intensive Systems*. ACM, 18:1–18:5.

[31] Kristof Meixner, Felix Rinker, Hannes Marcher, Jakob Decker, and Stefan Biffl. 2021. A Domain-Specific Language for Product-Process-Resource Modeling. In *26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE.

[32] Marcílio Mendonça and Donald D. Cowan. 2010. Decision-making coordination and efficient reasoning techniques for feature-based configuration. *Science of Computer Programming* 75, 5 (2010), 311–332.

[33] Stephan Mennicke, Malte Lochau, Julia Schroeter, and Tim Winkelmann. 2014. Automated verification of feature model configuration processes based on workflow Petri nets. In *SPLC*. ACM, 62–71.

[34] László Monostori. 2014. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP* 17 (2014), 9–13.

[35] William Motsch, Alexander David, Kirill Dorofeev, and Kathrin Gerber. 2021. Concept for Modeling and Usage of Functionally Described Capabilities and Skills. In *26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE.

[36] Sana Ben Nasr, Guillaume Bécan, Mathieu Acher, João Bosco Ferreira Filho, Nicolas Sannier, Benoit Baudry, and Jean-Marc Davril. 2017. Automated extraction of product comparison matrices from informal product descriptions. *Journal of Systems and Software* 124 (2017), 82–103.

[37] Kristin Paetzold. 2017. Product and Systems Engineering/CA* Tool Chains. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 27–62.

[38] Klaus Pohl, Günther Böckle, and Frank van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer.

[39] Clément Quinton, Daniel Romero, and Laurence Duchien. 2013. Cardinality-Based Feature Models with Constraints: A Pragmatic Approach. In *17th International Software Product Line Conference (SPLC '13)*. ACM, 162–166.

[40] Mikko Raatikainen, Juha Tiihonen, and Tomi Männistö. 2019. Software product lines and variability modeling: A tertiary study. *Journal of Systems and Software* 149 (2019), 485–510.

[41] Daniela Rabiser, Herbert Prähofer, Paul Grünbacher, Michael Petruzelka, Klaus Eder, Florian Angerer, Mario Kromoser, and Andreas Grimmer. 2018. Multi-purpose, multi-level feature modeling of large-scale industrial software systems. *Software and Systems Modeling* 17, 3 (2018), 913–938.

[42] Dieter Rombach. 2005. Integrated software process and product lines. In *Software Process Workshop*. Springer, 83–90.

[43] Marcello La Rosa, Wil M P Van Der Aalst, Marlon Dumas, and Fredrik P Milani. 2017. Business Process Variability Modeling: A Survey. *Comput. Surveys* 50, 1 (March 2017), 2:1–2:45.

[44] Emmanuelle Rouillé, Benoît Combemale, Olivier Barais, David Touzet, and Jean-Marc Jézéquel. 2012. Leveraging CVL to manage variability in software process lines. In *19th Asia-Pacific Software Engineering Conference*, Vol. 1. IEEE, 148–157.

[45] Miriam Schleipen, Arndt Lüder, Olaf Sauer, Holger Flatt, and Jürgen Jasperneite. 2015. Requirements and concept for Plug-and-Work. *at-Automatisierungstechnik* 63, 10 (2015), 801–820.

[46] Norbert Siegmund, Marko Rosenmüller, Martin Kuhlemann, Christian Kästner, Sven Apel, and Gunter Saake. 2012. SPL Conqueror: Toward optimization of non-functional properties in software product lines. *Software Quality Journal* 20, 3-4 (2012), 487–517.

[47] Jocelyn Simmonds, Daniel Perovich, María Cecilia Bastarrica, and Luis Silvestre. 2015. A megamodel for software process line modeling and evolution. In *ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 406–415.

[48] Maurice H. ter Beek, Axel Legay, Alberto Lluch-Lafuente, and Andrea Vandin. 2020. A Framework for Quantitative Modeling and Analysis of Highly (Re)configurable Systems. *IEEE Transactions on Software Engineering* 46, 3 (2020), 321–345.

[49] Tullio Tolio, Dariusz Ceglarek, Hoda A. ElMaraghy, Andreas Fischer, Shixin J. Hu, Luc Laperrière, Stepehen T. Newman, and József Váncza. 2010. SPECIES—Co-evolution of products, processes and production systems. *CIRP Annals* 59, 2 (2010), 672–693.

[50] Frank van der Linden, Klaus Schmid, and Eelco Rommes. 2007. *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering*. Springer Berlin Heidelberg.

[51] VDI/VDE 3682 2005. VDI/VDE 3682: Formalised Process Descriptions. Beuth Verlag.

[52] Roel J Wieringa. 2014. *Design science methodology for information systems and software engineering*. Springer.

### 5.3.4 Variability Modeling of PPR in CPPS Engineering

#### Citation

[137] **K. Meixner**, K. Feichtinger, S. Greiner, H. Marcher, H. S. Fadhlillah, R. Rabiser, and S. Biffl. Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering. *Journal of Systems and Software*, 211:112007, 2024. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2024.112007

#### Aim

The publication "Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering" introduces the EIPSE approach for holistically modeling and configuring models of variable CPPS assets and generating configured CPPS artifacts. The publication provides (i) an extension of the requirements for CPPS variability integration, (ii) the EIPSE approach to derive variability models for products, processes, and resources from a domain-specific description, (iii) in architecture and toolchain which automatically reduces the configuration space and allows to generate CPPS artifacts, (iv) a rigorous evaluation of the approach with four real-world use cases, including the generation of control code artifacts, and (v) an observational user study to collect feedback from engineers.

#### Contribution to the thesis

This publication contributes to the research goals CPPS knowledge model integration (**G3**.), PPR–DSL variability transformations (**G4**.), extended IRVM approach with CPPS variability exploration (**G1**., **G2**.), CPPS reuse method (**G3**., **G2**.), CPPS artifact configuration and generation (**G4**., **G5**.), multi-case study evaluation (**G5**.), and user study evaluation (**G5**.).

This publication contributes to **RQ1**., **RQ2**., and **RQ3**. by addressing the VDI 3695 measures of M1., *knowledge management* M2., *re-use* M3., *quality assurance* M4., *integration and seamless cooperation of disciplines* M5., and the SPL capabilities *efficient reuse* C1., *adaptability* C2., *variability management* C3., *enhanced quality and consistency* C4., and *facilitation of interdisciplinary collaboration* C5..

#### Abstract

CPPSs, such as *automated car manufacturing plants*, execute a *configurable* sequence of production steps to manufacture products from a product portfolio. In CPPS engineering, domain experts start with *manually* determining feasible production step sequences and resources based on *implicit knowledge*. This process is hard to reproduce and highly inefficient. In this paper, we present the EIPSE approach to derive variability models for products, processes, and resources from a domain-specific description. To automate the integrated exploration and configuration process for a CPPS, we provide a toolchain which automatically reduces the configuration space and allows to generate CPPS artifacts,

such as control code for resources. We evaluate the approach with four real-world use cases, including the generation of control code artifacts, and an observational user study to collect feedback from engineers with different backgrounds. The results confirm the usefulness of the EIPSE approach and accompanying prototype to straightforwardly configure a desired CPPS.

The Journal of Systems and Software 211 (2024) 112007

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

# Variability modeling of products, processes, and resources in cyber–physical production systems engineering[☆]

Kristof Meixner[a,b,*], Kevin Feichtinger[c], Hafiyyan Sayyid Fadhlillah[d], Sandra Greiner[e], Hannes Marcher[a], Rick Rabiser[d,f], Stefan Biffl[b]

[a] Christian Doppler Laboratory SQI, TU Wien, Favoritenstraße 9-11, Vienna, 1040, Austria
[b] Information Systems Engineering, TU Wien, Favoritenstraße 9-11, Vienna, 1040, Austria
[c] CRC 1608, KASTEL – Dependability of Software-intensive Systems, Karlsruhe Institute of Technology, Am Fasanengarten 5, Karlsruhe, 76131, Germany
[d] Christian Doppler Laboratory VaSiCS, Johannes Kepler University Linz, Altenberger Straße 69, Linz, 4040, Austria
[e] Software Engineering Group, Institute of Computer Science, University of Bern, Neubrückstraße 10, Bern, 3012, Switzerland
[f] LIT CPS Lab, Johannes Kepler University Linz, Altenberger Straße 69, Linz, 4040, Austria

ARTICLE INFO

Dataset link: https://github.com/tuw-qse/eipse

*Keywords:*
Variability modeling
Feature modeling
Decision modeling
Production process variability
Cyber–physical production system

ABSTRACT

Cyber-Physical Production Systems (CPPSs), such as *automated car manufacturing plants*, execute a *configurable* sequence of production steps to manufacture products from a product portfolio. In CPPS engineering, domain experts start with *manually* determining feasible production step sequences and resources based on *implicit knowledge*. This process is hard to reproduce and highly inefficient. In this paper, we present the Extended Iterative Process Sequence Exploration (eIPSE) approach to derive variability models for products, processes, and resources from a domain-specific description. To automate the integrated exploration and configuration process for a CPPS, we provide a toolchain which automatically reduces the configuration space and allows to generate CPPS artifacts, such as control code for resources. We evaluate the approach with four real-world use cases, including the generation of control code artifacts, and an observational user study to collect feedback from engineers with different backgrounds. The results confirm the usefulness of the eIPSE approach and accompanying prototype to straightforwardly configure a desired CPPS.

## 1. Introduction

Software Product Lines (SPLs) are "a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission developed from a common set of core assets in a prescribed way (Clements and Northrop, 2002)". ***Variability modeling*** is a crucial activity of SPL engineering. It captures the commonalities and differences of these software-intensive systems explicitly and manifests them in a variability model, such as a Feature Model (FM) (Kang et al., 1990) or Decision Model (DM) (Schmid et al., 2011). FMs focus on configuring products by respecting tree and cross-tree constraints, whereas DMs also allow configuring sequences of options, which, in particular, is useful to model process variability.

Cyber-Physical Production Systems (CPPSs), such as *automated car manufacturing plants*, use the latest information and communication technology to manufacture customized products with modern production techniques (Monostori, 2014). Each CPPS is built by assembling various hardware components (e.g., sensors and actuators) that are controlled by software to deliver one or more production system functionalities. Such systems execute a *configurable and flexible* sequence of production steps to manufacture products from a product portfolio, provoking variation in how to realize the CPPS (Järvenpää et al., 2019; Monostori, 2014). Hence, variability modeling for CPPSs faces the challenge of capturing multiple aspects (Fang et al., 2013; Galster et al., 2013) that reach from the variability of products to the sequence of production steps and employed production resources. Furthermore, engineers of different disciplines, such as mechanics, electrics, and software engineering, with different views on the planned CPPS collaborate to build it iteratively (Biffl et al., 2017). Due to this multidisciplinarity, ***separating the concerns is essential***, which is also true for variability modeling (Ananieva et al., 2016).

To build a CPPS, typically, CPPS engineers start with determining a feasible sequence of production process steps *manually* for the products in the portfolio (Lee, 1989). Then, they define which production

☆ Editor: Laurence Duchien.
* Corresponding author at: Christian Doppler Laboratory SQI, TU Wien, Favoritenstraße 9-11, Vienna, 1040, Austria.
*E-mail addresses:* kristof.meixner@tuwien.ac.at (K. Meixner), kevin.feichtinger@kit.edu (K. Feichtinger), hafiyyan.fadhlillah@jku.at (H.S. Fadhlillah), sandra.greiner@unibe.ch (S. Greiner), hannes.marcher@tuwien.ac.at (H. Marcher), rick.rabiser@jku.at (R. Rabiser), stefan.biffl@tuwien.ac.at (S. Biffl).

https://doi.org/10.1016/j.jss.2024.112007
Received 1 March 2023; Received in revised form 7 February 2024; Accepted 14 February 2024
Available online 24 February 2024

resources can execute these production process steps, examine the performance characteristics, and estimate the CPPS variant's construction cost. The production process steps and production resources are later used for designing and implementing CPPS artifacts, e.g., the control software. As the result originates from a completely manual process founded in *implicit domain knowledge*, resulting primarily *from experience and undocumented dependencies*, it is *hard and, most of the time, impossible to reproduce*. However, repetition of the planning may be necessary when a new product variant is introduced, "a frequent scenario in CPPS engineering" (Tolio et al., 2010). The manual configuration is also highly inefficient and tedious due to the large number of possible production sequences, challenging engineers to find practically feasible ones. Thus, this laborious, time-consuming, and error-prone activity calls for a methodology to automate and document the derivation of production sequence and resource configurations from a product configuration.

To address some of these challenges, we developed different automation facilities in previous work. We developed (i) the Product-Process-Resource Domain-Specific Language (PPR–DSL) (Meixner et al., 2021b) to model products with the required production processes and resources systematically; (ii) the TRAVART framework (Feichtinger et al., 2021) to transform engineering artifacts containing variability information into state-of-the-art variability models automatically; (iii) the Iterative Process Sequence Exploration (IPSE) (Meixner et al., 2022) approach that utilized these approaches to handle CPPS variability through SPL techniques and transformations of the PPR–DSL into a FM and a DM. However, the IPSE approach had *no automation and tool support to explore and configure process sequences*, and *did not include production resource modeling and configuration* and *artifact generation*.

In this paper, we extend the semi-automated IPSE process and aim to answer the following research questions:

**RQ1** *How can CPPS engineers be supported in modeling, exploring, and configuring the combined variability of products, production processes, and production resources, to generate corresponding CPPS artifacts?* The Extended Iterative Process Sequence Exploration (eIPSE) approach establishes a process that includes these artifacts and incorporates feedback loops that reflect the iterative development.

**RQ2** *How and to what extent can CPPS design be automated using variability modeling and CPPS concepts?* We provide the eIPSE tool architecture, which starts with the manually defined PPR–DSL and derives the remaining artifacts automatically.

In its first state (Meixner et al., 2022), the IPSE converted a given PPR–DSL into a FM to capture the variety of products and their parts, and a DM, to represent the production process sequences, using TRAVART (Feichtinger et al., 2021). We extend IPSE to derive a second kind of FM to capture the variability in production resources. Additionally, we utilize Cross-Discipline Constraints (CDCs) (Fadhlillah et al., 2022a,b) to express dependencies between variation points from those three variability models. Furthermore, we reduce the effort of manually configuring the resulting DM by offering an editor, which decreases the number of configuration options with each decision taken, and we empirically examine the gained automation. Thus, on top of our previous work, this paper contributes:

- The eIPSE approach with additional steps for production resource definition and configuration, artifact generation, and feedback loops.
- An extended prototype to assess the feasibility of the eIPSE approach, complemented with
  - support for transforming the PPR–DSL into a Resource FM and CDCs to elicit resource variability and dependencies among the different variability models
  - a novel DM editor for modeling and configuring DOPLER DMs, used in this context to represent and configure production process sequences
  - an automated reduction of the possible Process DM configuration based on the product configuration to lower the complexity of production process sequence configuration
  - support for resource configuration and control code artifact generation through integration with Variability for 4diac (V4rdiac) (Fadhlillah et al., 2022a,b) to automate the creation of control software.
- An empirical evaluation of
  - the feasibility of selecting an adequate CPPS variant in four *real-world case studies* (Meixner et al., 2021a)
  - applying the eIPSE approach in a new case study performed by engineers with heterogeneous backgrounds inexperienced in the approach, and thereby
  - the first exploration of the joint usage of feature and decision models for configuring CPPSs and creating CPPS artifacts.
- A report on the gathered insights from exploring production sequences in this highly automated way.

We postulate that the eIPSE approach, compared to the baseline of the traditional manual and hard-to-reproduce approach, (i) helps to externalize the implicit knowledge of engineers, (ii) reduces the effort of CPPS configuration, including the exploration of feasible production process sequences and (iii) separates the concerns of different engineering disciplines and, further, (iv) benefits the reproducibility of the configuration process. With these aspects, the approach directly addresses several criteria of the VDI 3695 guideline for optimizing industrial plant engineering (VDI, 2013; Jazdi et al., 2010). In particular, configuration and knowledge management, description languages, reuse, and the integration of disciplines. Additional material to this paper, such as the model artifacts and variability models, can be found online.[1]

The results of our empirical evaluation demonstrate that we can automate the subsequent configuration process of a CPPS by using variability models; a clear benefit compared to the manual assembly and exploration process. Our subjects spend the most time defining the PPR–DSL, while configuring the CPPS boils down to generating and configuring the variability models. However, this shows how much effort it is to externalize their implicit knowledge, which, on the other hand, supports engineers downstream the engineering process. Particularly, configuring the production process sequences through the DM benefits from our prototype, which displays only the decisions that can be made in the respective configuration step. Furthermore, the separation of concerns of splitting and linking the variability models for products, production processes, and production resources allows their configuration by respective experts. Consequently, the eIPSE process with tool support meets the expectations of automating and eliciting the configuration process of CPPSs, improving its reproducibility.

The remainder of this work is structured as follows: Section 2 summarizes the background on CPPSs engineering and variability modeling. Section 3 presents an illustrative use case. Section 4 describes our research methodology. Section 5 presents the eIPSE approach and corresponding tooling. Section 6 describes the evaluation of the eIPSE approach. Section 7 discusses the results and implications of the approach. Section 8 summarizes related work that aims at achieving similar goals and Section 9 concludes this work by providing an outlook on future work.

## 2. Background

This section provides background information on CPPS engineering and variability modeling.

[1] Additional material to eIPSE: https://github.com/tuw-qse/eipse.

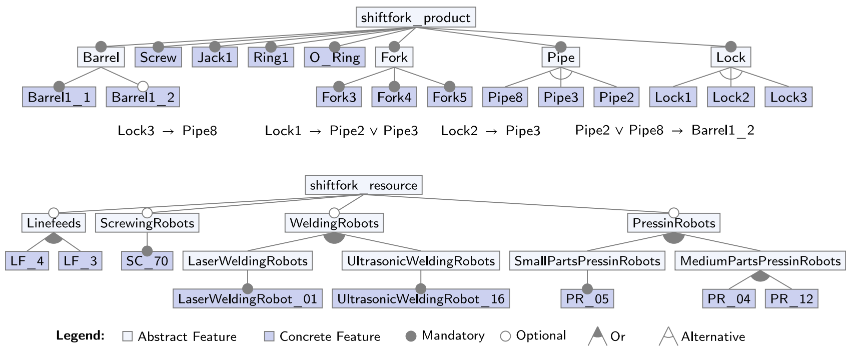


**Fig. 1.** FeatureIDE FMs (Meinicke et al., 2017) of product (top) and production resource (bottom) variability of the *shift fork* case study.

### *2.1. CPPS engineering*

CPPSs are next-generation production systems that interact autonomously with their environment, aiming for flexible production of customized products that build a product family (Meixner et al., 2020; Monostori, 2014). CPPSs are Cyber-Physical Systems (CPSs) (Gunes et al., 2014) with the purpose of manufacturing goods. CPPS engineering resides in a multidisciplinary environment that involves engineers from diverse disciplines, such as mechanical, electrical, and computer sciences (Biffl et al., 2017).

The engineers collaborate to create various artifacts representing aspects of the CPPS. For instance, they create technical documents such as text documents or spreadsheets for defining requirements or *bills of materials*. The engineers also create engineering artifacts, e.g., CAD drawings or AutomationML artifacts (IEC, 2014) to represent the CPPS's physical and functional design. Additionally, the engineers use domain-specific language defined by industrial standards, such as IEC 61499 (IEC, TC65/WG6, 2012), to implement the control software. Furthermore, several of these artifacts, e.g., type comparison matrices (TCMs), contain information on variability in the CPPS, such as product types or production processes (Feichtinger et al., 2020).

One prominent concept in CPPS engineering is the Product-Process-Resource (PPR) approach (Schleipen et al., 2015), which describes (i) **p**roducts and their parts, (ii) production **p**rocesses required to manufacture the products, and (iii) production **r**esources that execute the processes. The Formalized Process Description (FPD) (VDI, 2005) allows for modeling these PPR aspects in a visual (and partly formalized) model (cf. Fig. 2). Complementary, we contributed the PPR–DSL (Meixner et al., 2021a), which we use in this work, as a machine-readable PPR format that represents the variability and constraints among the PPR aspects.

### *2.2. Variability modeling*

Modeling variability explicitly is crucial for (software) product line engineering. In this work, two dimensions of variability play a role: CPPSs manufacture a product line of goods, such as families of cars, whereas CPPSs can also be configured in various ways. Furthermore, the sequence of the production steps involves dependencies between product parts and production resources and may vary in the estimated cost. The multidisciplinary nature of CPPS engineering calls for different views on the CPPS involving different variability models (Meixner, 2020; Meixner et al., 2019): feature models to capture structural variability of product parts and production resources, and decision models to capture the behavioral variability of production process sequences.

**Feature models** (Kang et al., 1990) elicit commonalities and differences in a feature tree and allow for defining cross-tree constraints, e.g., that one feature requires or excludes another. Given this model, we can perform a configuration by selecting and deselecting features while conforming to the expressed constraints. For instance, the FM on the top of Fig. 1 captures commonalities and differences of a shift fork (c.f., Section 3). The model consists of *mandatory* features representing product parts required in all variants, such as a *Screw* and the three types of forks. The model also contains *optional* features, such as *Barrel1_2*, and feature groups, such as the alternative *Pipe* group that allows selecting only one type of pipe. The FM on the bottom of Fig. 1 captures commonalities and differences of potential production resources, such as two *Linefeeds* that can be used to feed material into the CPPS.

**Decision models** (Schmid et al., 2011) are rooted in the Synthesis method (Campbell et al., 1990), which supports the reuse of processes and the necessary assets for configuring an application engineering process. DMs include only varying features and their constraints. For instance, the DOPLER approach (Dhungana et al., 2011) comprises DMs and asset models for defining variability in the problem space and reusable elements and their dependencies in the solution space, respectively. It maps assets onto the decisions, but decisions are unaware of the assets.

Table 1 represents an exemplary DM in tabular representation. A decision in a DM consists of a unique ID and a text describing the decision (usually a Question). These decisions are configured based on the specified Range. For instance, only one of the three locks can be selected in the enumeration decision *Lock*. Constraint/Rule and Visible/Relevant if relationships between decisions specify *(post-)conditions* and hierarchical or logical *(pre-)conditions*. The Visible/Relevant if relationship defines preconditions that need to be satisfied for the decision to be selectable. For instance, *InsertLock1* can only be selected if *Lock1* is selected. A visibility condition *false* (e.g., *InsertLock*) entails that the decision cannot be selected by developers but rather by selecting another decision that relies on this decision. Similarly, abstract features in FMs are automatically selected if one of their child features is selected. For instance, the selection of *InsertLock1* triggers the selection of *InsertLock*. These explicit dependencies among selected decisions reflect a configuration order, which models behavioral variability.

**CDCs** (Fadhlillah et al., 2022b,a) model the relationships among different types of variability models, likely employed by different (engineering) disciplines, as shown in Listing 1. A CDC identifies the involved variability model using the variability type and the unique id. For instance, *CDC1* in Listing 1 refers to the relation of a Product FM feature (shiftfork_product#Pipe2), which concerns product engineers,

**Table 1**
Excerpt of the generated DOPLER DM (Dhungana et al., 2011) representing the process variability of the *shift fork* case study in tabular notation (Schmid and John, 2004).

| **ID** | **Question** | **Range** | **Visible/Relevant** if | **Constraint/Rule** |
|---|---|---|---|---|
| Pipe | Which Pipe types? | Pipe2 \| Pipe 3 \| Pipe8 | false | |
| Barrel1_2 | Install Barrel1_2? | true \| false | false | |
| Lock | Which Lock types? | Lock1 \| Lock2 \| Lock3 | false | Lock1 $\Longrightarrow$ Pipe = Pipe2 $\vee$ Pipe = Pipe3<br>Lock2 $\Longrightarrow$ Pipe = Pipe3<br>Lock3 $\Longrightarrow$ Pipe = Pipe 8 |
| InsertPipe | Install InsertPipe? | true \| false | false | |
| InsertPipe2 | Install InsertPipe2? | true \| false | Pipe == Pipe2 | InsertPipe2 $\Longrightarrow$ InsertPipe |
| InsertLock | Install InsertLock? | true \| false | false | |
| InsertLock1 | Install InsertLock1? | true \| false | Lock == Lock1 | InsertLock1 $\Longrightarrow$ InsertLock |
| InsertLock2 | Install InsertLock2? | true \| false | Lock == Lock2 | InsertLock2 $\Longrightarrow$ InsertLock |
| InsertBarrel1_2 | Install InsertBarrel1_2? | true \| false | Barrel1_2 | |
| PressBarrel1_2 | Install PressBarrel1_2? | true \| false | Barrel1_2 & InsertBarrel1_2 & InsertPipe | |
| … | … | … | … | … |

```
CDC1) shiftfork_product#Pipe2 => shiftfork_process#Pipe2
CDC2) shiftfork_product#Pipe2 => shiftfork_process#InsertPipe2
CDC3) shiftfork_process#WeldLock =>
    shiftfork_resource#WeldingRobots
CDC4) shiftfork_product#Lock1 => shiftfork_product#Pipe2 ||
    shiftfork_product#Pipe3;
```

Listing 1: Excerpt of CDCs (Fadhlillah et al., 2022b) of the *shift fork* case study.

to the Process DM decision (shiftfork_process#Pipe2), which concerns production process engineers, to model dependent processes in the decision model. Similarly, *CDC3* models the relation of the *WeldLock* process to the *WeldingRobot* production resource that concerns production resource engineers.

## 3. Motivating case study

This section presents the shift fork case study (Meixner et al., 2021a), one example of a CPPS, based on which we illustrate our contribution in the following sections. Shift forks are part of a manual transmission in a car.[2] They shift the cuffs with two forks along the pipes to their correct position to connect the transmission's gears. Typically, a single shift fork moves a particular cuff for two gears, e.g., the first and the third gear. This design results in a product portfolio of four shift fork variants required for a particular type of car transmission.

In the first CPPS engineering phase, so-called *basic engineers* analyze the product portfolio that the CPPS should produce (Meixner et al., 2021a), which represents a classical (mechanical) product line with common and varying features. The engineers examine various artifacts, such as CAD drawings or product prototypes, to identify the commonalities and differences of the product line (Meixner et al., 2021a). Furthermore, the engineers examine the required partial production steps, such as joining the fork and the pipe of the shift fork by welding them together (Meixner et al., 2021a). Fig. 2 shows a section of such a production step sequence for a single product in VDI 3682 notation (VDI, 2005) (cf. Section 2), where, in the first step, a *Pipe* (product) *is inserted* (process) into the CPPS using a *Linefeed* (resource).[3] In this step, they often disassemble the prototypes and put

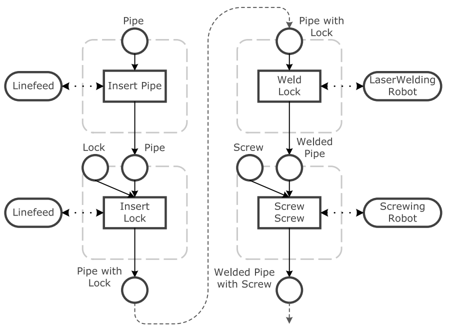


**Fig. 2.** Excerpt of a PPR model for the production steps of a shift fork in VDI 3682 notation (VDI, 2005).

them "back together" (Lee, 1989). The engineers aim to design feasible production processes for the requested product line. They complement these designs with potential basic CPPS designs and corresponding rough cost estimates (Meixner et al., 2021a).[4]

Some product parts require a specific production sequence. For instance, the pipe must be available to mount forks onto the pipe. Still, several degrees of freedom to assemble a particular product remain, which must be resolved either in engineering or operation. Traditionally, the engineers design feasible process sequences, e.g., in tools like DelMia,[5] based on implicit knowledge, by using heterogeneous and partial data representations. The engineers use the sequences to reason about process characteristics, such as the production duration. However, as the decisions are typically undocumented, a change to the CPPS, which occurs frequently, may cause redesigning the entire CPPS from scratch (Paetzold, 2017; Meixner et al., 2021a). Therefore,

[2] Exemplary figure of *shift forks*: https://w.wiki/3DCf.

[3] In contrast to the figure, the engineers first plan the single process steps, e.g., *insert pipe* in an isolated way.

[4] If customers accept a CPPS design and cost estimate, the artifacts of *basic engineering* are handed over to the different disciplines of *detail engineering*. As the time of a cost estimate and its acceptance can be wide apart, the engineers in *detail engineering* need to reiterate over those designs, ideally, reuse, and detail them.

[5] DelMia: https://www.3ds.com/products-services/delmia/.

```
1   Product "Pipe": { name: "Abstract Pipe", isAbstract: true }
2   Product "Pipe2": { name: "Pipe 2",
3     implements: ["Pipe"],
4     excludes: [ "Pipe3", "Pipe8" ]
5   }
6
7   Product "Lock": { name: "Abstract Lock", isAbstract: true ,
8     requires: ["Pipe"]
9   }
10  Product "Lock1": { name: "Lock 3",
11    implements: ["Lock"], excludes: [ "Lock2", "Lock3" ]
12  }
13
14  Process "InsertPipe": { name: "InsertPipe", isAbstract: true }
15  Process "InsertPipe2": { name: "InsertPipe2",
16    implements: ["InsertPipe"],
17    inputs: [ {productId: "Pipe2"} ],
18    outputs: [ {OP1: {productId: "Pipe2"}} ],
19    resources: [ { resourceId: "Linefeeds" } ]
20  }
21  Process "WeldLock": { name: "WeldLock", isAbstract: true ,
22    requires: [ "InsertLock", "InsertPipe", ... ],
23    inputs: [ {productId: "Lock"}, {productId: "Pipe"} ],
24    outputs: [ {OP2: {productId: "ForkProduct"}}],
25    resources: [ {resourceId: "WeldingRobot"} ]
26  }
27  Process "WeldLock1": { name: "WeldLock1",
28    implements: [ "WeldLock" ], inputs: [ "Lock1" ]
29  }
30
31  Resource "WeldingRobot": { name: "WeldingRobot",
32    isAbstract: true }
33  Resource "LaserWeldingRobot_01":{ name: "LaserWeldingRobot_01",
34    implements: [ "LaserWeldingRobots" ]
35  }
36
37  Constraint "C1": {
38    definition: "Lock1,Pipe2,Pipe3 -> Lock1 implies Pipe2 OR Pipe3"
39  }
```

Listing 2: Excerpt of a PPR model for the *shift fork* case study in PPR–DSL (Meixner et al., 2021b).

making this knowledge more explicit is a decade-old challenge in CPPS engineering (Drath et al., 2008; Drath, 2009).

To elicit their domain knowledge, CPPS engineers can use the PPR–DSL (Meixner et al., 2021b) to represent PPR aspects and their constraints. Listing 2 shows an excerpt of a PPR–DSL file that defines *shift fork* product parts (lines 1–12), the atomic production process steps and variants (lines 14–29), the production resources (lines 31–35), and the constraints between these PPR aspects (lines 37–39). The entire PPR–DSL file is available online.[1]

For instance, Line 1 defines the abstract Product *Pipe* that builds the central part of a *shift fork*. The Product *Pipe2*, defined in the following line, represents a concrete pipe and excludes another variant of a pipe (i.e., *Pipe3* and *Pipe8*). Similarly, *InsertPipe* in Line 14 represents an abstract Process which is implemented, for instance, by the Process *InsertPipe2*. The PPR–DSL further enumerates its input and output products, i.e., *Pipe2*, and required production resources. Examples of production resources, which can be abstract or concrete, are stated in Lines 31–33. The last lines of the excerpt, Lines 37 to 39, add the Constraint *C1* which defines that the *Lock1* implies the presence of either *Pipe2* or *Pipe3*.

Thus, the PPR–DSL describes the products and their parts, the required production process steps, and the production resources of the shift fork case study precisely. Particularly, it elicits the variability and dependencies among the three aspects.

## 4. Methodology

To develop the eIPSE approach, we applied the Design Science methodology (Hevner, 2007; Hevner et al., 2008). This methodology aims to solve problems by covering the design and investigation of artifacts that define, above others, methods and technical solutions in a problem context to improve something in that context (Hevner

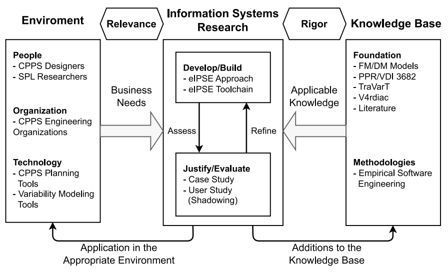


**Fig. 3.** Design Science methodology (Hevner, 2007; Hevner et al., 2008) for this work.

et al., 2008; Wieringa, 2014). In this work, the method is the iterative design and investigation of the eIPSE approach in the context of CPPS engineering. Fig. 3 shows the Design Science methodology adapted to this work, with the *relevance* cycle on the left, the *design* cycle in the middle, and the *rigor* cycle on the right.

Our previous work (Feichtinger et al., 2020, 2021; Meixner et al., 2022, 2021b) builds the *grounding* in the *rigor cycle*. Furthermore, we aim to address literature gaps, in particular, the integration of different variability models of CPPSs (Krüger et al., 2017) including their behavior (Fang et al., 2013; Galster et al., 2013) and the separation of concerns between engineers of different disciplines (Ananieva et al., 2016). Based on discussions with stakeholders of our industry partners and our previous work, we obtain additional requirements for an integrated CPPS variability modeling approach (cf. Section 5.1) for the *relevance cycle*.

We iteratively perform the following tasks for the *design cycle* and the *develop/build activity* (cf. Section 5): First, to address the identified requirements, we extend our former approach with respective steps, resulting in the eIPSE approach, and investigate which steps can be further automated (cf., Section 5.2). We lay out the architecture of the eIPSE toolchain to illustrate a prototype implementation (cf., Section 5.3). We integrate a DOPLER DM into the state-of-the-art technology and develop the remaining transformation operations for converting the PPR–DSL into three variability models. We develop a *DM Editor* as a prototype for modeling and configuring the Process DM and design the automated reduction of the Process DM configuration based on the configuration of the Product FM. Then, we adapt V4rdiac to use the distinct configurations of the Product FM and Process DM for generating artifacts. In this way, we establish the eIPSE toolchain to support the steps of the eIPSE approach (cf. Section 5). Furthermore, we aim to separate the concerns of *basic engineers* and the disciplines involved in *detail engineering*, particularly product design, production process engineering, and production resource engineering.

For the *evaluate activity* of the *design cycle*, i.e., the evaluation of the approach and prototype, we rely on a three-fold approach. First, we applied the eIPSE approach to a set of previously published case studies (Meixner et al., 2021a) from the CPPS domain to investigate the feasibility of the approach (cf. **EQ1** and Section 6.3). Second, we conducted an observational user study (Wohlin et al., 2012; Runeson et al., 2012; Singer et al., 2008) with six engineers to provide evidence on the perceived usability of the eIPSE approach and learn about its usage. Therefore, we requested study subjects to perform the eIPSE process on a newly introduced case study (cf. **EQ1** and Section 6.4). A detailed description of the user study can be found in the appendix, cf. Appendix. Third, we investigate the configuration space's reduction for the production sequences of one particular case study using the eIPSE approach (cf. **EQ2** and Section 6.5). Finally, we examined the generation of parameterized CPPS artifacts for a particular CPPS configuration for each of the four previously mentioned case studies (cf. **EQ3** and Section 6.6). These measures complete the *design cycle*.

We discuss our findings and lessons learned and accompany them with additional material[1] and a demonstration video[6] as additions to the knowledge base. Furthermore, we briefly describe measures to assess the practical impact of the application in CPPS engineering. These measures aim to reach a broad audience in academia and industry and complete the *rigor* and *relevance cycle* (cf. Section 7).

## 5. Adopting variability modeling for CPPS process exploration and configuration

This section details employing variability modeling for exploring configuration options in deriving a functional CPPS design. Firstly, Section 5.1 states the requirements for a CPPS production process exploration and production resource configuration approach as elicited from industrial needs in previous work. Inferred from these requirements, Section 5.2 describes increasing automation in CPPS planning and configuring through the Extended Iterative Process Sequence Exploration (eIPSE) approach. To assess the feasibility of the eIPSE approach and to follow the Design Science methodology, Section 5.3 presents our prototype implementation, respectively.

### 5.1. Requirements for eIPSE

To gather requirements for automating the configuration of CPPSs, we conducted interviews with industrial partners in previous work (Meixner et al., 2022) on which we build in this article. The resulting requirements **R1**-**R3** mainly motivate knowledge representation and tool support as follows:

**R1. Production Variability Exploration.** The approach shall collect variability knowledge from CPPS engineering (artifacts) that is required to explore production process sequence options efficiently. The approach must incorporate new knowledge from product line evolution, such as changes of the products that the CPPS manufactures or process simulation and optimization iteratively.

**R2. Product & Process Variability Representation.** The knowledge representation shall describe

(i) the products, which form a product line,
(ii) the atomic production process steps, which form a process line, both with their dependencies and CDCs in an industrial variability artifact, e.g., using the PPR–DSL (Meixner et al., 2021b), and
(iii) variability models of the product and process line in state-of-the-art variability models (e.g., FMs or DMs).

**R3. Variability Transformation & Configuration.** The IPSE method and tool support shall (i) automate transformations of the variability represented in the industrial artifacts into state-of-the-art variability models and (ii) provide guidance through product and process variability model exploration and configuration supported by tools, such as the FeatureIDE (Meinicke et al., 2017) or DOPLER tool (Dhungana et al., 2011) to better support multidisciplinary engineering.

However, requirements R1–R3 only focus on exploring and configuring products and processes without considering production resources. Thus, based on previous work and discussions with stakeholders, we defined two additional requirements (**R4** and **R5**) that address the necessity to include the production resources in the IPSE.

**R4. Resource Variability Representation.** After exploring feasible production sequences, the engineers' goal is to ''search'' for suitable production resources, e.g., welding robots, that are able to execute the production steps (cf. Section 3). Therefore, the eIPSE knowledge representation shall model a product line of potential production resources that can execute respective production process steps. The product line should be represented in an industrial variability artifact, e.g., a variability model. Additionally, the eIPSE method shall respect the production resource variability model when being transformed and configured (**R3**). The production resource variability model shall be transformed into state-of-the-art variability models and adequate tool support shall guide configuration.

**R5. Cross-Discipline Constraint Representation.** The CPPS engineering process expects engineers to link the developed concepts to a system design, e.g., that a production process requires a particular type of welding robot. To complement the overall eIPSE knowledge representation, we must be able to describe dependencies (include/exclude relations) between variability knowledge across different PPR concepts.

These requirements reflect the need of engineers to continuously incorporate additional product requirements and subsequently assign production processes and resources to the CPPS design. Furthermore, they represent the engineers' demand to obtain a holistic overview of a CPPS's variability in their preferred engineering artifacts and models that can be better computed, e.g., for satisfiability, such as state-of-the-art variability models. Beyond that, the requirements build the foundation for the automation of an integrated CPPS variability modeling approach.

### 5.2. The extended IPSE approach

In prior work (Meixner et al., 2022), we introduced the linear IPSE approach, which utilizes state-of-the-art variability models to enable the systematic and reproducible exploration of potential production process sequence and resource configurations based on a product configuration. These variability models need to support *structural* variability to represent the hierarchical structure of products and *behavioral* variability to represent the potential sequences of process steps. The tool support for the original IPSE approach allows for *exploring process sequences* manually, but *does not* include the modeling and configuration of *production resources* and the *artifact generation*. To address these issues and satisfy the additional requirements **R4** and **R5**, we utilize a FM to represent resource variability and CDCs (Fadhlillah et al., 2022b) to represent one or more dependencies across PPR concepts of different disciplines. Here, each CDC is a propositional logic constraint based on the variation points defined in Product FM, Process DM, and Resource FM (cf. Listing 1). To this end, we extend our previous work with the eIPSE approach and toolchain.

*Overview.* In this article, we contribute the eIPSE approach, which extends the *linear* IPSE approach with

(i) automated transformations from the PPR–DSL to the Resource FM and the CDCs for linking the variability models,
(ii) an automated reduction of the Process DM configuration based on the Product FM configuration and the *tool-supported* process exploration that guides engineers,
(iii) the automated reduction of the Resource FM configuration and its *tool-supported* configuration,
(iv) the generation of CPPS artifacts, such as control code artifacts,
(v) and feedback loops to respect its iterative character.

This way, eIPSE aims to support the disciplines of functional product design, production process engineering, and production resource engineering. Fig. 4 illustrates the resulting eIPSE process. Steps with dashed contours in Fig. 4 were carried over *as-is* from the IPSE approach, steps with solid contours were adapted, and steps with solid contours and in dark green were newly introduced in the eIPSE approach. The process consists of ''human'' tasks conducted by engineers with tool support (persona with cog icon) and automated tasks (cog icon). eIPSE provides automation support by utilizing the eIPSE tool. While the eIPSE tool performs the automated tasks entirely, it supports CPPS engineers during the ''human'' tasks that require access to models.

The eIPSE process starts with the definition of the PPR element variants of the desired CPPS by engineers in the PPR–DSL and ends

[6] eIPSE demonstration video: https://youtu.be/eoNNDOusXKA.

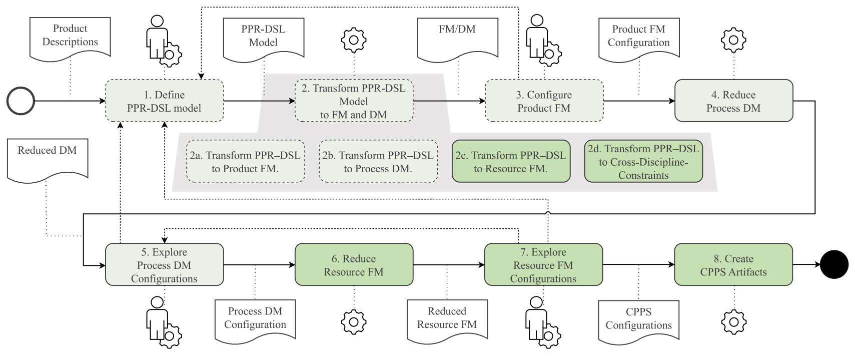

Product Descriptions
PPR-DSL Model
FM/DM
Product FM Configuration
1. Define PPR-DSL model
2. Transform PPR-DSL Model to FM and DM
3. Configure Product FM
4. Reduce Process DM
Reduced DM
2a. Transform PPR–DSL to Product FM.
2b. Transform PPR–DSL to Process DM.
2c. Transform PPR–DSL to Resource FM.
2d. Transform PPR–DSL to Cross-Discipline-Constraints
5. Explore Process DM Configurations
6. Reduce Resource FM
7. Explore Resource FM Configurations
8. Create CPPS Artifacts
Process DM Configuration
Reduced Resource FM
CPPS Configurations

**Fig. 4.** (Human & automated) eIPSE process steps for exploring production process steps based on a product configuration (updated steps have solid contours, novel steps, additionally a darker color).

in creating the artifacts for a configured CPPS automatically. The PPR–DSL is automatically transformed into three variability models (Step 2a-c) and their respective CDCs (Step 2d). The variability models are configured by engineers and automatically and subsequently configured for the next configuration step (Steps 3–7).

In contrast to the IPSE process, the eIPSE process splits the second step into sub-steps to demonstrate the various variability models that the eIPSE tool creates based on the PPR–DSL. It creates an additional Resource FM to represent the production resources (Step 2c) and derives and elicits the CDCs among the three variability models (Step 2d). The reduction of the Process DM configuration (Step 4) is automated. The exploration and configuration of the Process DM (Step 5) is integrated with *state-of-the-art* technology and supported by the eIPSE tool. Subsequently, in Step 6, the additionally created Resource FM configuration is automatically reduced based on the decisions taken (i.e., the configuration of the Process DM), and next, configured by the CPPS engineer with the eIPSE tool. Finally, Step 8 creates the control code for the configured production resources to operate the CPPS. The following descriptions highlight the added and adapted steps compared to the original IPSE and explain the eIPSE tool.

*Details.* First, CPPS engineers model product parts and product variants and identify atomic production steps and production resources to manufacture these products based on analyzing the product descriptions and production requirements in Step 1. In this way, they iteratively define a PPR–DSL model (cf. Listing 2). In practice, engineers typically select and adapt such atomic process steps and, particularly, production resources from an artifact catalog.

After defining the PPR–DSL, in Step 2, the eIPSE uses transformation technology, such as TRAVART (Feichtinger et al., 2021), to create the variability models automatically. In contrast to the IPSE process, which derived a Product FM and a Process DM only, this second step transforms the PPR–DSL into three variability models with shared CDCs (cf. requirement *R3* & *R5*). It creates

(i) a Product FM to represent the product variants, their parts, and structure (cf. upper FM in Fig. 1),
(ii) a Process DM to represent the atomic process steps and their dependencies, representing partial behavior of the CPPS, and product decisions not shown to engineers during the configuration but required for the satisfiability calculation of the model (cf. Table 1),
(iii) a Resource FM to represent the hierarchical structure of production resources (cf. lower FM in Fig. 1), and
(iv) CDCs capturing the dependencies between the Product FM, the Process DM, and the Resource FM in propositional logic (cf. Listing 1).

The following steps configure and reduce the possible configuration spaces of the variability models. The CPPS engineer starts with configuring the Product FM in Step 3, resulting in a valid configuration according to the FM. For instance, an engineer selects the *Pipe2* and *Lock1* in the Product FM configuration (cf. Fig. 1). Based on the configuration, the eIPSE tool in Step 4 automatically reduces the subsequent Process DM configuration to exclude decisions that are unnecessary to produce the configured product. For each selected feature in the Product FM configuration, the configuration value for the corresponding decision in the Process DM is set to true. For instance, for the selected *Pipe2* feature, the eIPSE tool will set the configuration value of the *Pipe2* product decision to true. For all product features that are not selected, the configuration value of the respective decision remains `false`. The configuration values in the Process DM set in this way affect which process decisions are visible during the configuration, e.g., decision *InsertPipe2* in Table 1, due to its visibility conditions.

In this reduced Process DM configuration, in Step 5, a CPPS engineer can explore the remaining process decisions iteratively and interactively with the eIPSE tool. Based on a constant evaluation, the tool only displays the process steps that are feasible during the configuration stage according to the visibility conditions. Based on the internal constraints of the Process DM, the eIPSE tool sets the subsequent configuration values. Furthermore, the eIPSE tool stores and visualizes a queue representing the sequence of the currently taken decisions. The final sequence of the configured decisions represents the desired production sequence. This production sequence can be used to define and optimize a valid production process model that is executed on the CPPS.

In Step 6, the eIPSE tool automatically reduces the Resource FM configuration (cf. requirement *R3*) by considering the Process DM configuration of Step 5. This results in a partial Resource FM configuration, where the configuration possibilities are a valid subset of production resource configurations for a particular product and process sequence configuration. For each selected decision in the Process DM configuration from Step 5, the eIPSE tool considers the CDCs for the production processes and resources. If a process requires a type of production resource, the configuration value for the corresponding features in the Resource FM is preselected. For instance, if the *WeldLock1* decision is part of the configured process sequence, the *WeldingRobot* group is selected.

Based on the Resource FM, in Step 7, a CPPS engineer can configure the desired production resources with the eIPSE tool. As the Resource FM is pre-configured by the Process DM configuration (Step 5), the configuration of the Resource FM only contains the production resources to be used in the production process.

In the final step, Step 8, engineering and operation artifacts should be generated from the combined configuration of the Product FM, the Process DM, and the Resource FM. For instance, IEC 61499 (IEC, TC65/WG6, 2012) or AutomationML (Drath, 2021) code can be parameterized and generated from reusable artifacts. This step aims to increase the reusability of artifacts and decrease artifact creation time to increase engineering productivity. To support this automation, an eIPSE tool requires additional mechanisms for generating CPPS artifacts. Therefore, CPPS engineers need to decide on a *variability mechanism* (Apel et al., 2013), such as *Delta Modeling* (Schäfer et al., 2021; Zhang et al., 2016), to implement the shared and the varying CPPS artifacts. In Delta modeling, engineers create elements, such as stubs, templates, and lines of code, that represent the base implementation, in our case, particular CPPS artifacts such as control code. Then, they define Delta models comprising modifications to enrich the base implementation. These operations can modify the base implementation or add or remove code (cf. Listing 3). During artifact generation, a generator parameterizes and combines this base implementation and the variable code parts, i.e., the Deltas, depending on the variability models' configuration.

*Summary.* The result of the eIPSE process is a set of configurations as well as engineering and operation artifacts, such as control code for a possible design of a CPPS or plan for the assembly of a concrete product. This set of configurations can be used, for instance, to generate a layout of the CPPS, optimize it, and initiate its operation.

### 5.3. eIPSE Toolchain Architecture

This section describes our implementation of an eIPSE tool. In particular, it presents the architecture of our prototype and further implementation details. Fig. 5 depicts this architecture with its main components, which consist of (i) the PPR–DSL to define, read, and evaluate PPR models (eIPSE, Step 1), (ii) TraVarT to transform PPR–DSL models into FMs and DM (Step 2), (iii) the FeatureIDE to configure the Product FM (Step 3), (iv) the eIPSE DM editor to read or define, manipulate, and configure DMs (Steps 4 & 5), and (v) V4rdiac to read, manipulate, and configure models with CDCs (Steps 6 & 7) and generate IEC 61499 code based on Delta models (Clarke et al., 2015) (Step 8). Compared to our previous work, we provide additional automation support with extended TraVarT transformations for production processes and resources in the PPR–DSL, introduce the novel eIPSE DM editor, and integrate V4rdiac. To this end, in Fig. 5, unchanged components are depicted with dashed contours, and novel or adapted components are depicted with solid contours.

*PPR–DSL environment.* The PPR–DSL environment is realized as a standalone Java application (cf. Fig. 5, top left in yellow). Its main components are the PPR Model (based on an extended VDI 3682 model (VDI, 2005)), the PPR Command Line Interface (CLI), the PPR Parser, and the PPR Evaluator for constraints. The parser reads the PPR File, which includes the products, production process steps, production resources, and constraints via the CLI and builds the PPR Model. The constraints are mapped onto (recursive) SQL queries and evaluated using a PostgreSQL database. This strategy also allows the use of aggregation functions, such as the sum or average of attribute values, over a PPR hierarchy and an easy integration with industrial standards.

Currently, the tool support for the PPR–DSL comprises a set of snippets and code completion functions implemented in SublimeText.[7]

[7] SublimeText: https://www.sublimetext.com/.

**Table 2**
Mapping table of PPR–DSL onto Variability Model Elements.

| PPR-DSL | | FM and DM elements |
|---|---|---|
| Unit | Product | Feature with attribute, Decision |
| | Process | Decision |
| | Resource | Feature with attribute |
| Properties | Name | – |
| | Abstract | Abstract feature |
| | implements | Feature tree if only one other unit and feature attribute otherwise feature attribute |
| | children | Feature tree |
| | requires | Implies constraint |
| | excludes | Excludes constraints |
| Constraints | Not | Not constraint |
| | And | And constraint |
| | Or | Or constraint |
| | Implies | Implies constraint |

The snippets provide stubs of PPR aspects with their attributes (cf. Listing 2 lines 1, 14, 31, and 37) that engineers can easily fill in. The code completion allows the recommendation and autocompletion of keywords and the aspects' IDs and names, such as the resource *Linefeeds* in Listing 2 line 19, so engineers can quickly find existing aspects. However, the text editor does not highlight incorrectly written keywords or whether engineers deleted aspects that are used in other aspect definitions. To this end, the development of an improved editor that supports features, such as code highlighting and missing aspects, using Eclipse XText[8] for better integration into the ecosystem, is ongoing.

*TRAVART environment.* TraVarT (Feichtinger et al., 2021) is a plugin-based variability model transformation environment (cf. top right of Fig. 5).[9] The TraVarT core plugin is implemented in Java and uses the Universal Variability Language (UVL) (Sundermann et al., 2021) as the pivot model, building on the current parser implementation.[10] For each supported variability model, a plugin needs to be implemented.

Such a plugin must provide functions for reading and writing the supported variability model. Furthermore, one has to specify Transformation Operations, which transform the supported variability model into the pivot model and vice versa. These operations are usually built upon a mapping table between the supported variability model and the pivot model and then implemented in Transformation Algorithms (Feichtinger et al., 2022a; Feichtinger and Rabiser, 2020a,b). Table 2 shows a mapping between the PPR–DSL and FM and DM aspects. Optionally, a plugin can implement a configuration Model Sampler to enable further testing of the resulting models.

Available plugins for TraVarT, including FeatureIDE FMs (Meinicke et al., 2017) and DOPLER DMs (Dhungana et al., 2011), implement transformation operations (Feichtinger et al., 2021) that map their concepts to the UVL (Sundermann et al., 2021) and vice versa (Feichtinger and Rabiser, 2020b). For instance, a decision in the DOPLER DM is mapped to a feature in the UVL (Sundermann et al., 2021). Also, a rule in the DOPLER DM is mapped to either a feature property (mandatory), the FM tree, or a constraint (Feichtinger and Rabiser, 2020b). In the opposite direction, the hierarchy of the FM tree is captured via the visibility conditions of the DOPLER DM. To support the needs of the eIPSE tool, we extended the DM Plugin by a new writer. This writer creates a file conforming to the syntax of the DM editor (see paragraph below), a propositional logic syntax for constraints and visibility conditions.Moreover, we iteratively extended

[8] Eclipse XText: https://www.eclipse.org/Xtext/.
[9] TraVarT: https://github.com/SECPS/TraVarT.
[10] UVL Parser: https://github.com/Universal-Variability-Language/uvl-parser.

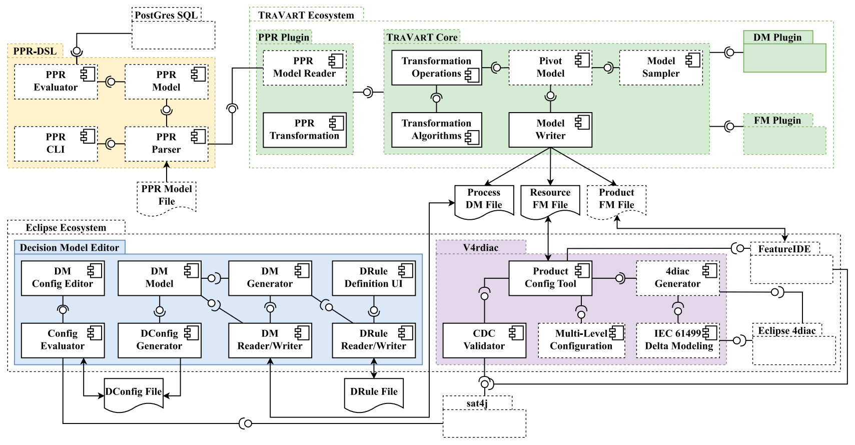


**Fig. 5.** Architecture of the eIPSE toolchain in UML component diagram notation (novel and updated components are depicted with solid contours).

the existing PPR--DSL Plugin (Feichtinger et al., 2022a) and transformation operations (cf. Table 2) to transform PPR–DSL processes into a Process DM and PPR–DSL resources into a Resource FM. Specifically for the Process DM, the plugin creates a single Boolean decision for each process in the PPR–DSL with its input products and required preceding processes as visibility conditions. For instance, the plugin creates a decision *InsertPipe2* in the Process DM in Table 1 from the *InsertPipe2* process in Listing 2. Furthermore, it creates a visibility condition that fires if, for decision *Pipe*, the value *Pipe2* is selected. Subsequently, it creates a constraint that selects the abstract decision *InsertPipe*. Abstract processes are transformed to Boolean decisions with a visibility condition false, to be implicitly selected by those constraint rules. Further, the transformations create a feature for each PPR–DSL resource and derive its properties (e.g., whether the feature is abstract or mandatory) from the respective PPR–DSL properties. For example, for the *WeldingRobot* (cf. Listing 2), an abstract feature is created in the Resource FM, defining a group of welding robots. For the *KUKA_KR_Agilus*, a concrete feature is generated in the *WeldingRobot* group. Each group of PPR–DSL resources is converted into an *OR*-group because at least one of the grouped resources will have to be selected if these resources are necessary to produce the configured product. Finally, the plugin derives a list of CDCs by connecting the features and decisions from the resulting variability artifacts, i.e., Product FM, Process DM, and Resource FM. For instance, the necessary PPR–DSL resources for a given process result in a CDC between the respective process decision in the Process DM and the respective resource features in the Resource FM (cf. CDC2 in Listing 1).

*Eclipse ecosystem.* The Eclipse Ecosystem[11] is a cross-platform, open-source integrated development environment (IDE) for (software) engineering in different languages and domains. The IDE provides a plugin system that allows the dynamic exchange of components. Thus, it allows combining different independent software components into a suitable toolchain, in our case, for variability modeling.

*FeatureIDE environment.* The FeatureIDE (Meinicke et al., 2017)[12] is the current de-facto-standard open-source plugin for feature-oriented software development. It supports several FM types, among others, graphical FMs modeling and textual variability modeling using UVL (Sundermann et al., 2021). Furthermore, the FeatureIDE allows for configuring FMs and validating them through sat4j.

*Eclipse 4diac™.* Eclipse 4diac™[13] (Zoitl et al., 2010) is an open-source Eclipse-based tool for developing IEC 61499-based (IEC, TC65/WG6, 2012) control software for CPPSs.

*DM editor environment.* The eIPSE DM Editor (bottom left of Fig. 5) is inspired by the DOPLER DM editor (Dhungana et al., 2011). The latter was developed for an industry partner and is a closed-source tool. Instead, our eIPSE DM Editor (cf. Fig. 5) is an open-source Eclipse plugin, compatible with the latest version of FeatureIDE. Our DM Editor offers the following key features: (i) the import of DOPLER DMs, used here to import the Process DM, (ii) the creation of DRule files, where each file represents a single decision, (iii) the generation of DOPLER DMs from those DRule files, (iv) the generation of DConfig configuration files for DOPLER DMs, and (v) the configuration of DOPLER DMs via those DConfig configuration files, used here to explore and configure the Process DM.

DRule files exhibit a similar structure as the DOPLER DM in Section 2.2, supporting a propositional logic syntax for constraints and visibility conditions. Currently, we limited the syntax to types that can be translated into common SAT solvers, in this context sat4j, to enable the DOPLER DM's configuration validation and integration into the toolchain. On user-triggered generation, the DM Generator component maps the DRule files onto an internal DM Model. From this DM Model, the DM Writer writes a DM File with the specific syntax. In our case, this model and the corresponding files contain the decisions for the production process steps that need to be selected to create a suitable process step sequence. However, this model could also be used for other types of decisions.

The DConfig Generator component generates a configuration for a DM in a DConfig File. It uses the DM Reader that reads the DM model file created either by the DM Generator or by TRAVART.

[11] Eclipse Foundation: https://www.eclipse.org.

[12] FeatureIDE: https://github.com/FeatureIDE/FeatureIDE.

[13] Eclipse 4diac™: https://www.eclipse.org/4diac.

The DM Config Editor reads the generated DConfig File and presents configuration options in a configuration view. In this view, users can configure the DM model, which is validated in the background by the Config Evaluator. The selected sequence of decisions is stored in the DConfig File for further processing, for instance, to export or visualize the production process sequence.

*V4rdiac.* V4rdiac (Fadhlillah et al., 2022b) is a, currently closed source, multidisciplinary variability management approach for CPPS variability realized as an Eclipse plugin (cf. bottom right of Fig. 5). The eIPSE tool uses several V4rdiac components for generating customer-specific control software based on the selected products, production processes, and production resources. Specifically, the eIPSE tool uses the components IEC 61499 Delta Modeling, Multi-Level Configuration, CDC Validator, Product Config Tool, and the 4diac Generator.

Eclipse 4diac™ is used to develop the base implementation of the IEC 61499 control software. Beyond that, the IEC 61499 Delta Modeling component is used to implement the Deltas for the control code artifacts (cf. Listing 3). Afterward, a mapping between the base implementation artifacts and the Deltas needs to be established. In our case, CPPS engineers define an additional attribute in the PPR–DSL, which specifies the location of corresponding Deltas, e.g., as URI (cf. Listing 4 Line 8). The Multi-Level Configuration is used to define a step-wise configuration in which different variability models can be configured separately. In our case, the Product FM and Process DM with their configurations and the Resource FM are loaded into the component and ordered into this sequence. During the configuration of the Resource FM, which allows the configuration of multiple production resources for a production process, the CDC Validator ensures that the selected configurations are valid according to the CDCs. The Product Config Tool component displays the configuration user interface for the Resource FM. After the Resource FM configuration in the Product Config Tool, the eIPSE process is finished with a valid configuration. The eIPSE tool requires an artifact generator that evaluates the mapping between PPR–DSL attributes and CPPS artifacts to remove, combine, and build CPPS artifacts given a set of selected products, production processes, and production resources. The IEC 61499 Delta Modeling component is linked to Eclipse 4diac™ and the 4diac Generator to generate the IEC 61499 control code. The 4diac Generator then generates the control software for the production resources in IEC 61499. For a detailed description of Delta modeling for control software, we refer to (Fadhlillah et al., 2023a,b).

The eIPSE tool aims to support a straightforward integration of different CPPS artifact generators and formats, such as IEC 61499 (IEC, TC65/WG6, 2012) or AutomationML (Drath, 2021) code.

## 6. Evaluation

This section describes the evaluation of the eIPSE approach and prototype. Section 6.1 presents the evaluation questions to be answered. The subsequent sections (cf. Sections 6.2–6.6) first present the general setup of the evaluation activities, followed by describing the concrete setup and the results.

### 6.1. Evaluation questions

To evaluate the eIPSE approach and prototype, we address the research questions (cf. Section 1) and stated requirements (cf. Section 5.1) with the following evaluation questions.

**EQ1** *Is it feasible to apply the eIPSE approach*

(a) in different real-world case studies?
(b) by engineers from heterogeneous backgrounds?

We postulate that the eIPSE approach facilitates the externalization of knowledge, reduces the effort of CPPS modeling and configuration through automation, including the exploration of feasible production process sequences, and benefits the reproducibility of the configuration process. To examine EQ1, we applied our approach to different real-world case studies from industry (Meixner et al., 2021a). Beyond that, we shadowed (Shull et al., 2007) subjects with different backgrounds to investigate the utility of our approach, i.e., whether it is feasible enough to be used in CPPS engineering. Therefore, we measured their efforts as "time spent" when applying the eIPSE approach step-wise and configuring a CPPS as a notable factor of CPPS engineering optimization (VDI, 2013) and for future investigation. Furthermore, we collected feedback from the subjects in post-task discussions. For both investigations, we take the engineers' hard-to-reproduce and manual approach as a baseline for optimization, where engineers employ mostly implicit domain knowledge to design and configure CPPSs.

**EQ2** *By how much can using eIPSE reduce the number of decisions needed to configure a production process sequence for a CPPS?*

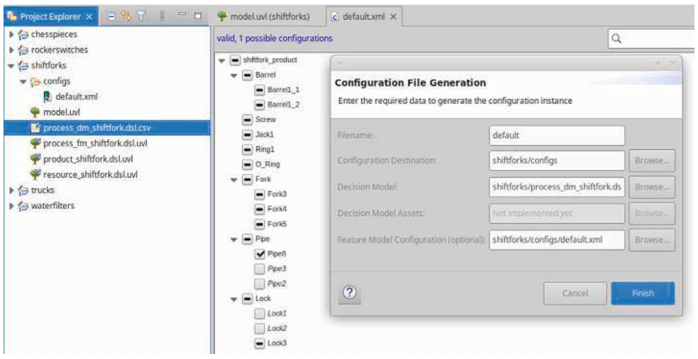
The eIPSE approach is grounded on the hypothesis that the configuration of a single product and the formulation of pre- and post-conditions for process steps can significantly reduce the configuration space for production process sequences. By reducing the configuration space, the users are guided to only configuring necessary process steps, which is essential when configuring commercial and/or industrial software (Hubaux et al., 2012). Therefore, the eIPSE approach uses DOPLER DMs due to the concept of visibility conditions that allows for a subsequent unfolding of configuration options in contrast to FMs. We address EQ2 by comparing the entire configuration space of a Process DM with the configuration space for the reduced Process DM resulting from using eIPSE by utilizing combinatorics. In particular, we focus on a subsequently created production process sequence for a particular product configuration of the *shift fork* case study (Meixner et al., 2021a).

**EQ3** *Can the eIPSE tool chain generate consistent CPPS control software code?*

The logical consequence of exploring the process sequences and configuring the production resources for a particular product configuration is generating artifacts that represent various CPPS aspects. In our work, we are currently focused on one type of CPPS artifact, i.e., IEC 61499 (IEC, TC65/WG6, 2012) control software. We address this question by preparing multiple valid combinations of selected products, production processes, and production resources. We use each valid combination to generate IEC 61499 control software variants using our toolchain. Then, we evaluate the consistency of the control software code by verifying whether the elements related to the selected product, production process, and production resources exist in the generated control software.

### 6.2. General evaluation setup

For the evaluation, we installed the eIPSE toolchain on one of the author's notebooks. This is due to the company policies of some of our evaluation subjects from industry, they are not allowed to install any additional software, including our eIPSE toolchain. Our setup included (i) SublimeText as a text editor to manipulate the PPR–DSL including a "cheat sheet" for its syntax (eIPSE, Step 1), (ii) TRAVART with the PPR–DSL as the library to transform the PPR–DSL models to the required variability models (Step 2), (iii) Eclipse with FeatureIDE (Step 3), the DM Editor (Steps 4 &5), and V4rdiac to manipulate and configure the variability models (cf. Step 6 &7), and to generate the CPPS artifacts (Step 8) showing them with 4diac as plugins. We used this setup in the sessions to investigate the evaluation question **EQ1** +**EQ3**.



valid, 1 possible configurations

Configuration File Generation

Enter the required data to generate the configuration instance

Filename: default
Configuration Destination: shiftforks/configs
Decision Model: shiftforks/process_dm_shiftfork.ds
Decision Model Assets: Not implemented yet
Feature Model Configuration (optional): shiftforks/configs/default.xml

**Fig. 6.** After configuring the Product FM in Step 3 of the eIPSE process (background) for the *shift fork* case study (Meixner et al., 2021a), the reduced Process DM configuration is created in Step 4 of the eIPSE process using the eIPSE prototype's wizard (front).

**Table 3**
Statistical data on the PPR–DSL artifact and the generated Product FM, Process DM, Resource FM, and CDCs.

| **Case study** | **PPR-DSL artifact** | | | | | **Product FM** | | | | | | **Process DM** | | | **Resource FM** | | | | | **CDCs** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | #Products | #Product$_{comp}$ | #Processes | #Resources | #Constraints | #Features | #Constraints$_{xor}$ | #Constraints$_{or}$ | #Constraints$_{tree}$ | Tree height | #Configs | #Decisions | #Constraints | #Constraints$_{vis}$ | #Features | #Constraints$_{xor}$ | #Constraints$_{or}$ | #Constraints$_{tree}$ | Tree Height | #Rules |
| Truck | 12 | 7 | 13 | 3 | 30 | 8 | 1 | 0 | 0 | 2 | 4 | 20 | 32 | 20 | 5 | 0 | 1 | 0 | 2 | 15 |
| Shift fork | 24 | 19 | 36 | 16 | 52 | 20 | 2 | 0 | 10 | 2 | 4 | 55 | 27 | 53 | 17 | 0 | 8 | 0 | 3 | 35 |
| Rocker switch | 46 | 33 | 59 | 13 | 63 | 34 | 0 | 0 | 20 | 2 | 44 | 92 | 26 | 92 | 14 | 0 | 5 | 0 | 3 | 63 |
| Water filter | 46 | 37 | 36 | 13 | 108 | 38 | 8 | 0 | 54 | 3 | 10 | 73 | 127 | 73 | 14 | 0 | 5 | 0 | 3 | 43 |

Additionally, our evaluation subjects are distributed in diverse locations. To solve this limitation, we utilize Zoom's *Screen Sharing* and *Remote Control* features to use the eIPSE toolchain remotely. In this way, we can use the same machine specification for every subject using the eIPSE toolchain in the evaluation. Furthermore, we can record those Zoom sessions for later analysis.

### 6.3. Application of eIPSE to case studies

This evaluation activity addresses **EQ1 a** by investigating the applicability of the eIPSE approach to real-world case studies from industry and by measuring the resulting design and configuration space.

*Setup.* In prior work (Meixner et al., 2021a), we introduced four real-world CPPS case studies: *truck*, *shift fork*, *rocker switch*, and *water filter*, modeled their products in the PPR–DSL and implemented TRAVART transformation operations. In Meixner et al. (2022), we extended the *shift fork* model with processes and added the corresponding TRAVART transformation.

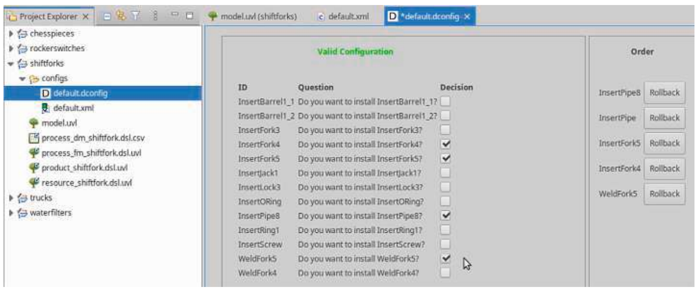
For the evaluation activity, the author most familiar with the PPR–DSL and the particular CPPSs modeled the remaining atomic process steps and resources (eIPSE, Step 1) for each case study. The author utilized engineering artifacts of the respective CPPSs. In the *shift fork* case study, the author most familiar with TRAVART iteratively improved the existing and newly implemented transformation operations, sustaining the research methodology. These adaptations primarily concerned the hierarchy and grouping of PPR aspects but did not change the nature of the CPPS designs.

The two authors alternately applied the remaining steps of the eIPSE approach (Steps 2 to 7) to each of the four case studies. Figs. 6 to 8 show (i) the configuration view of the Product FM in Step 3 (background) and the wizard to create the Process DM configuration in Step 4 (front), (ii) the DM Editor and a step in configuring the Process DM in Step 5, and (iii) the Resource FM configuration in Step 7 for the *shift fork* case study using our eIPSE prototype.

According to the feedback loops shown in Fig. 4, the two authors incorporated changes during the evaluation activity iteratively to correct errors in the PPR–DSL model, such as missing exclusion or grouping constraints and to omit errors in the generated variability models. To this end, we also used the eIPSE approach to *validate* the PPR models of the case studies. A third author took notes and acted as a referee to minimize bias during the evaluation activity, which could have been introduced by the familiarity of the other two authors with the case studies. Additionally, during the transformations, we ran automated statistics for each case study to obtain various metrics for the PPR–DSL file and resulting variability models. We built on previously defined metrics (Feichtinger et al., 2022a), such as the size of the models, their constraints, and configuration space. We summarize the resulting statistics in Table 3 and explain them below. We used the notes by the third author and the collected statistics to further improve the implementation of our prototype (cf. Section 4).

*Results.* Our evaluation showed that we were able, with the feedback loops, to apply the eIPSE approach in the four selected case studies. While we had to adapt the PPR–DSL models iteratively throughout the process, we were able to configure reasonable production process sequences and production resources for a particular product configuration. This indicates that applying eIPSE to industrial CPPS product lines is feasible.

As a supplementary result, we gathered metrics of the variability models resulting from the transformation, listed in Table 3. As a representative of the case studies, we explain the *shift fork* case study in detail. The first category summarizes the metrics of the *PPR–DSL artifact*. For the *shift fork* case study, there are 24 product definitions



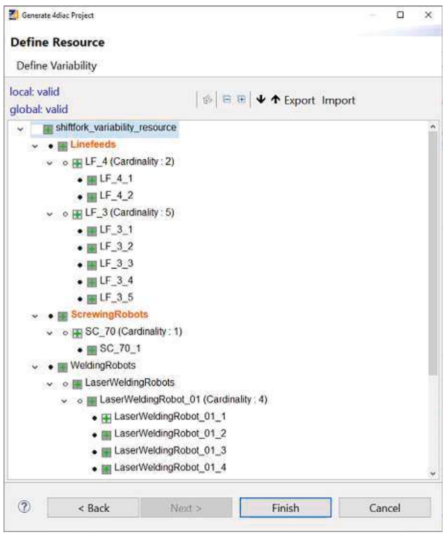
Valid Configuration

| ID | Question | Decision |
|---|---|---|
| InsertBarrel1_1 | Do you want to install InsertBarrel1_1? | ☐ |
| InsertBarrel1_2 | Do you want to install InsertBarrel1_2? | ☐ |
| InsertFork3 | Do you want to install InsertFork3? | ☐ |
| InsertFork4 | Do you want to install InsertFork4? | ✓ |
| InsertFork5 | Do you want to install InsertFork5? | ✓ |
| InsertJack1 | Do you want to install InsertJack1? | ☐ |
| InsertLock3 | Do you want to install InsertLock3? | ☐ |
| InsertORing | Do you want to install InsertORing? | ☐ |
| InsertPipe8 | Do you want to install InsertPipe8? | ✓ |
| InsertRing1 | Do you want to install InsertRing1? | ☐ |
| InsertScrew | Do you want to install InsertScrew? | ☐ |
| WeldFork5 | Do you want to install WeldFork5? | ✓ |
| WeldFork4 | Do you want to install WeldFork4? | ☐ |

Order

| | |
|---|---|
| InsertPipe8 | Rollback |
| InsertPipe | Rollback |
| InsertFork5 | Rollback |
| InsertFork4 | Rollback |
| WeldFork5 | Rollback |

**Fig. 7.** During the configuration of the production steps in the Process DM in Step 5 of the eIPSE process, a suitable production process sequence for the *shift fork* case study (Meixner et al., 2021a) is defined. The eIPSE prototype provides a *rollback* option (right-hand side) for systematic process sequence exploration (Meixner et al., 2022).



Define Resource

Define Variability

local: valid
global: valid

```
shiftfork_variability_resource
  • Linefeeds
    o LF_4 (Cardinality : 2)
      • LF_4_1
      • LF_4_2
    o LF_3 (Cardinality : 5)
      • LF_3_1
      • LF_3_2
      • LF_3_3
      • LF_3_4
      • LF_3_5
  • ScrewingRobots
    o SC_70 (Cardinality : 1)
      • SC_70_1
  • WeldingRobots
    o LaserWeldingRobots
      o LaserWeldingRobot_01 (Cardinality : 4)
        • LaserWeldingRobot_01_1
        • LaserWeldingRobot_01_2
        • LaserWeldingRobot_01_3
        • LaserWeldingRobot_01_4
```

**Fig. 8.** The configuration of the Resource FM in Step 7 of the eIPSE process for the production resources necessary to execute the configured production process steps of the Process DM in Step 5 of the *shift fork* case study (Meixner et al., 2021a).

with 19 product component definitions included, 36 atomic process step definitions, 16 resource definitions, and 52 constraints. The next set of metrics concerns the generated *Product FM*. We measured 20 features resulting from the set of 19 product components plus the root feature. The constraints in the PPR–DSL artifact were transformed into 2 xor groups and 10 cross-tree constraints. The reduced complexity compared to the PPR–DSL artifact results from the number of constraints necessary to describe an alternative group in the PPR–DSL. The 4 possible configurations for the products in the Product FM represent exactly the 4 shift fork types produced in the real-world CPPS. The same holds for the 4 trucks in the *truck* case study. However, the Product FM is underconstrained, resulting in more possible configurations than the modeled final product types in the PPR–DSL of the *rocker switch* (44 configurations vs. 12 product types) and the *water filter* (10 configurations vs. 8 product types) case studies. In the generated *Process DM*, we measured 55 decisions, consisting of 19 product component decisions, of which 4 were abstract, 15 were concrete, and 36 processes from the PPR–DSL artifact. Those 19 decisions are used to pre-configure the Process DM for the process exploration (cf. Step 4 in Section 5). The table shows the large number of decisions (55), rules/constraints (27), and visibility constraints (53) compared to the constraints in the PPR model (52). The generated *Resource FM* contains 17 features derived from the 16 defined resources in the PPR–DSL model plus the root feature. The features are grouped in or groups based on the constraints defined in the PPR–DSL. The last category shows the generated *CDCs*, which are 35 derived CDC rules for the *shift fork* case study. The table shows that the PPR–DSL model requires fewer constraints (52) than the variability models combined; 2+10 for the Product FM, 27+53 for the Process DM, 8 for the Resource FM, and 35 CDCs.

### *6.4. Application of eIPSE by different engineers*

This evaluation activity addresses **EQ1 b**. In a user study, we investigate how much effort engineers from heterogeneous backgrounds inexperienced in the eIPSE approach spend for each step of applying eIPSE by shadowing them (Shull et al., 2007). We report on their effort, experience, and perceived usefulness. For a detailed description of the user study, we refer to Appendix.

*Setup.* For this evaluation activity, we introduce the new *chess piece* case study originating from the TU Wien pilot factory.[14] The product line consists of six chess piece types with a body and an aluminum base with either one or two carved reamings. The body and the base are joined via threaded rods of two lengths.

Five subjects applied the eIPSE approach to this case study. An overview of the subjects and their domain can be found in Table 4. The first and the second columns of the table state the subject and whether the subjects are engineers (*E*) or researchers (*R*) and their domain. Subject S1 is an engineer at an industry partner in the field of high-speed CPPS automation with a background in mechanical engineering (*ME*). Two subjects (S2 & S3) are engineers from an industry partner in the automotive domain with a background in mechanical engineering. Subject S4 is a senior systems engineer (*SE*) in the CPPS domain from a research collaboration. Subject S5 is a computer science researcher with a mechanical engineering background. All subjects know the principles of the PPR concept.

All subjects conducted the evaluation activities in individual Zoom sessions after the evaluation activity for **EQ1 a** (cf. Section 6.3) under

[14] Pilotfabrik TU Wien: https://www.pilotfabrik.at.

**Table 4**
"Time spent" by engineers with different backgrounds for eIPSE to the *chess piece* case study. E…Engineer, R…Researcher; ME…Mechanical engineering, SE…Systems engineering, CS…Computer science.

| **Subject** | | **Activity** | | | | | |
|---|---|---|---|---|---|---|---|
| | | Intro | Engineering | | Configuration | | |
| Participant | Profession | | Definition | Updates | Product FM | Procss DM | Resource FM |
| S1 | E,ME | 10 m | 60 m | 30 m | 3 m | 5 m | 1 m |
| S2 | E,ME | 13 m | 70 m | 15 m | 2 m | 7 m | 3 m |
| S3 | E,ME | 9 m | 72 m | 3 m | 2 m | 5 m | 4 m |
| S4 | E,SE | 6 m | 62 m | 4 m | 1 m | 4 m | 1 m |
| S5 | R,ME | 9 m | 63 m | 9 m | 6 m | 9 m | 3 m |
| Summary (avg) | | 9 m | 65 m | 12 m | 3 m | 6 m | 2 m |

the supervision of at least two authors.[15] If the subjects had questions concerning the PPR–DSL, the eIPSE approach, or the toolchain, these authors provided tips and support. These authors gave hints on how to model aspects of the product line once a subject got stuck and asked for help. We also measured the time the subjects took to execute each step of the evaluation activity and eIPSE process. We aimed to understand better where subjects need more automation support and where we can further improve the eIPSE approach. Furthermore, we wanted to compare the use of the eIPSE process with a control group that performs the same tasks without the eIPSE process.

The subjects had to model the chess piece types and their parts, reasonable atomic production steps, and production resources in a PPR–DSL model (eIPSE, Step 1). In this evaluation, it is not necessary for each subject to model the full version of the chess piece product line. We mainly focused on ensuring that the subjects grasp the overall idea of using our PPR–DSL and gain feedback from them. Furthermore, we wanted to get an indication of the relation between the tasks for engineering and configuration. The subjects then use their resulting PPR–DSL model to generate the Product FM, the Process DM, and the Resource FM using the TRAVART CLI (Step 2). Then, the subjects configured the Product FM (Step 3) and loaded the configuration into the DM Editor to create a reduced DM configuration file (Step 4) (cf. Fig. 6). Afterward, the subjects explored the process sequence and configured the Process DM (Step 5). Lastly, they configured the Resource FM in V4rdiac, which reduced the model based on the configured Process DM (Step 6 &7). If the subjects thought it was necessary to improve their variability models, they used the feedback loops (cf. Fig. 5) to improve their product line.

After the subjects had completed this evaluation task, we asked them for their experience with and feedback on the eIPSE approach as well as its perceived usefulness. We analyzed the notes taken by comparing them and finding evidence for the usefulness and benefits of the approach, its steps, and potential limitations and improvements.

*Results.* Table 4 presents the times spent on this evaluation activity. The third column states the time to introduce the eIPSE approach and the *chess piece* product line. The fourth and fifth column concern the domain engineering phase, showing the time spent to define the chess piece product line as a PPR–DSL model. The fifth column shows the time the subjects used to update the product line after an initial configuration. This update corresponds to the feedback loops of the eIPSE approach (cf. Fig. 4 dotted arrows). The last three columns present the times spent during the configuration phase (i.e., application engineering) indicating the efforts for configuring the Product FM in the FeatureIDE (Meinicke et al., 2017), the Process DM in the eIPSE DM Editor, and the Resource FM in V4rdiac (Fadhlillah et al., 2022b). The last row summarizes the rounded average time of each step in minutes.

The introduction to eIPSE and the *chess piece* case study took, on average, 9 min, [min. 6 mins, max. 13 mins]. Spending on average 65 minutes, [60 mins, 72 mins], the subjects spent most of the time on *defining the product line.* While most of the subjects did not model the entire product line of the six chess pieces with all the required production processes and resources, they could all grasp the concepts of the approach and continue with the configuration. *Updating the product line* according to the feedback loops took the subjects on average 12 minutes [3 mins, 30 mins] depending on how much they updated their models. For the initial PPR–DSL model transformation to the variability models and their configuration, we used the subjects' PPR–DSL models. However, as the subjects often did not model all products, production process steps, and production resources for timely reasons, after the first investigation of their generated variability models, we switched to a prepared *chess piece* PPR–DSL model to ensure a likewise feedback regarding the configuration with the eIPSE approach.

The *CPPS configuration* in this approach was fast for the Product FM and the Resource FM (both have avg. 3 minutes). The Process DM configuration took the evaluation subjects on average 6 minutes, [4 mins, 9 mins], slightly longer than the configuration of the other variability models. One reason might be that users novel to the approach do not have enough experience in ordering process steps in a meaningful way. Additionally, it seems that more domain knowledge is required to decide which process steps seem reasonable to be ordered in a particular way. Thus, subjects used more time to experiment with different sequences before finalizing and deciding on a specific one.

In post-task discussions, we gathered feedback on the approach from each subject. Overall, the eIPSE approach was well received. Impressions were that the approach

- "*is very helpful and the toolchain works great*". (S2)
- "*makes sense from an engineer's perspective*". (S1/S4)
- "*makes the knowledge about the production sequence explicit*" (S2/S3/S4)
- "*is straightforward*". (S1/S4)
- "*allows for a more economic and optimized design of the CPPS*". (S3)

Subject S3 stated that "*the process digitalization is a great idea that can improve reuse of existing configurations*" and it is "*easy to understand and use*". Several subjects stated that it "*supports the reproducibility of process selection*". S4 confirmed that the separation of concerns through "*modeling relations from different discipline perspectives*" is important. S2 meant that "*the often rigid integration structures of large companies might render the approach better suited for small and medium companies*".

On the constructive side, the subjects also pointed out some issues and suggested several improvements. Most subjects noted that the approach "*definitely requires better tool support to use it efficiently*", such as "*better tool feedback*" and "*a better overview of PPR concepts*" using, for example, "*low code approaches*".

Concerning the PPR–DSL for modeling the *chess piece* product line, the subjects stated that

[15] The time frame was limited to roughly two hours for industrial subjects.

- it “*provides the means for better reuse*” of engineering artifacts (S4) and
- was “*straightforward and easy to use once the syntax is clear*”. (S2)
- “*the PPR–DSL is great because it is not as complex as, e.g., SysML*”. (S2)

Nevertheless, three subjects noted that the PPR–DSL “*has a steep learning curve*”. Several subjects commented that the PPR–DSL “*is sometimes redundant and partially confusing*”, for instance, because a “product” is the same as a material,[16] and that the “*difference between the requires and children relation is unclear*”. Such limitations and the “*several times missing overview*” can make the PPR–DSL “*as-is error-prone for larger models*”. This concerns, for instance, “*the definition of constraints in the model with a large number of products/processes*”. While the “*cheat sheet was of great use*”, the PPR–DSL “*should be simplified to be usable for engineers*” and “*redundancies should be omitted*”. A suggestion was also to “*provide more examples and best practices*” and “*improve the documentation*” for the PPR–DSL. Furthermore, the subjects desired the “*introduction of parallel processes*”, “*definition of transport in the CPPS*”, and “*use of libraries*”. We aim to enhance the PPR–DSL and its tool support to address these comments in the future.

Feedback on the transformation to the variability models was positive: “*extremely fast and happens in the background once syntax errors are fixed*”. However, the transformation “*may cause iterative loops of updating the PPR–DSL*”. We argue that these feedback loops are intended and should be used by engineers to improve their PPR models. We also argue that more experienced users and product designers presumably define better PPR models.

Feedback on the configuration was mainly positive:

- it “*provides an easy configuration*”. (S1/S4)
- “*having the dependencies explicitly transferred into the configurable models helps to build feasible production sequences*” (S5)
- “*the experimentation with different process sequences through simulation is a good idea*”. (S3)
- the exploration makes sense particularly with “*digital twins and asset administration shells for simulation and provides additional value with the modeling of the process relations*”. (S4)

All subjects also provided several remarks to improve our eIPSE approach and toolchain. In particular, Subject S5 stated that “*cost/risk assessment would be nice to further enhance and evaluate the process sequences*”. Several subjects also stated that “*the toolchain requires better integration*”. For instance, one subject mentioned that “*executing the process using the eIPSE toolchain requires a lot of preparatory steps, which could be reduced*”. Asking for clarification, the subject explained that copy-pasting the necessary files between the process steps should be enhanced. We argue that the current state of our toolchain, not the eIPSE process itself, caused this statement. In future work, we aim to integrate all involved toolchain tools into a single tool environment, such as Eclipse.

### 6.5. Reduction of the process configuration space

This evaluation activity addresses **EQ2** by investigating the reduction of the configuration space of a Process DM.

*Method.* To measure the reduction of the configuration space, we utilize combinatorics. Considering the sequence of process steps, the Process DM configuration follows a permutation, an arrangement of elements in a specific sequence. Additionally, only visible process steps (visibility condition is true) can be configured in the Process DM configuration. The formula to calculate permutations, where $n$ denotes the total number of visible atomic process steps and $r$ denotes the number of steps occurring only in combination, is defined as:

$$P(n,r) = \frac{n!}{(n-r)!} \tag{1}$$

To this end, we compare the entire configuration space of a Process DM, where process steps can be combined arbitrarily, with the configuration space for the reduced Process DM resulting from a configured product using eIPSE (cf. Step 3 and Step 4). As a representative, we examine a subsequently created production process sequence in the *shift fork* case study.

*Results.* In this evaluation, we perform a simulation to configure the *shift fork*’s Product FM (cf. Fig. 1 and in Fig. 6) by selecting the features *Pipe2* and *Lock1* (Step 3). This selection results in a valid configuration representing a single shift fork product. Step 4 reduces the configuration options of the Process DM to this product where only the investigation of this product’s process sequence is progressed.

Consequently, the visibility conditions and configuration values of *Pipe2* and *Lock1* are set `true` in the Process DM (cf. Table 1). The Process DM configuration is further reduced by setting the visibility conditions of the mandatory process steps to `true`. Additionally, any alternative group within the Process DM is reduced by setting the visibility conditions of only one of the available options to `true`, leaving the engineer to select only truly variable process steps. For instance, the *InsertLock1* visibility condition is set to `true` and, thus, the *InsertLock2* and *InsertLock3* visibility conditions are set to `false`.

The overall Process DM configurations in the *shift fork* case study comprise 21 process step decisions and the 3 alternative process step group decisions when *considering* the *Pipe2* and *Lock1* product configuration AND the constraints between the decisions. In the case of the production process sequence exploration (Meixner et al., 2022), $r = n$ since we can combine all visible production process steps in the particular configuration step. Furthermore, arbitrarily combining the atomic process steps visible to developers at any time results in $n!$ permutation possibilities for $n$ decisions. Thus, the entire permutation space comprises $24! = 6 \cdot 10^{23}$ possible process step sequences.

Transforming the pre- and postconditions of the production processes in the PPR–DSL to visibility conditions in the Process DM enables the subsequent unfolding of the production process steps for creating a production process sequence. The shift fork’s product configuration allows 11 process steps, e.g., *InsertPipe2* and *InsertLock1*, with no visibility conditions that build the starting point of the Process DM configuration. Furthermore, the shift fork’s product configuration allows 4 process steps with visibility conditions related to the 11 previous steps, e.g., *InstallLock1*,. Following the visibility conditions, the eIPSE approach reduces the configuration of the Process DM to *five* consecutive steps with 11, 4, 6, 2, and 1 remaining production process steps. In each step, engineers had to decide in which sequence the currently possible production process steps have to be executed. Consequently, the eIPSE approach reduced the set of possible process sequences to a minimum of $11! + 4! + 6! + 2! + 1! \approx 39.9 \cdot 10^6$ possible sequences – a reduction of about $10^{17}$ sequence options.

Even though many sequence configuration options remain, the reduction is significant and helps to reduce the cognitive level of deciding on a valid and feasible sequence. However, the exploration of optimized production process sequences still demands additional knowledge and training from the engineers, which is a complementary activity to introducing the eIPSE workflow.

### 6.6. Generation of control software

This evaluation activity addresses **EQ3** by investigating how to create parts of the control software for a particular CPPS configuration as an example for a CPPS artifact.

*Setup.* The author, most familiar with V4rdiac, defined the necessary Delta files to generate IEC 61499 control code in 4diac as a CPPS

[16] We note that the VDI 3682 standard uses the term *product* for materials as well as complex composite products.

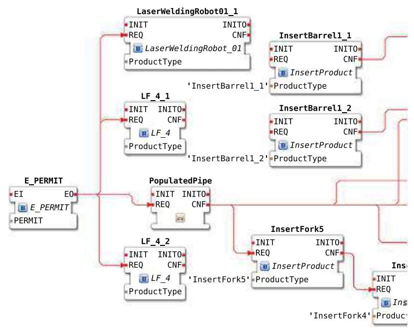


**Fig. 9.** Excerpt of a generated IEC 61499 control software with production process and resource function blocks for a shift fork configuration.

artifact of the four case studies during Step 8. In close cooperation with the authors of the first evaluation activity, the link between the processes and resources in the PPR–DSL to the Delta files was realized using a newly introduced PPR–DSL attribute.

```
delta DLock1;
uses ShiftForkCaseStudyApp;
{
  <Remove> NetworkElement name=InsertLock1;
  <Remove> NetworkElement name=WeldLock1;
  <Remove> NetworkElement name=E_REND_WeldLock1;
  <Add> FB name=UltrasonicWeldingRobot16
    type=UltrasonicWeldingRobot_16;
  <Add> EventConnection UltrasonicWeldingRobot16.CNF
    PopulatedPipe.REQ;
}
```

Listing 3: Snippet of an IEC 61499 Delta model for the *shift fork* case study.

An example of a Delta model can be seen in Listing 3. Each Delta model may define a unique identifier by using the delta keyword. The uses keyword sets the context for the modification. A <Remove> and an <Add> operation define which element will be removed from or added to the CPPS artifacts, respectively.

Next, the author introduced a new attribute deltaFile to the PPR models of the four case studies and used it to refer to the particular Delta files (cf. Listing 4) from the processes and resources (eIPSE, Step 1). Afterward, the author re-ran the TraVarT transformations (Step 2) to generate the attributes in the variability models. Then, the author went through the configuration and reduction steps of the eIPSE approach according to the setting of the first evaluation activity (cf. Section 6.3) covering Steps 3 to 7 as input for Step 8. In Step 8, the author triggered the generation of parts of control software in IEC 61499 in V4rdiac.

*Results.* Our toolchain successfully generated an IEC 61499 control software based on a set of selected products, production processes, and production resources. Additionally, all elements in the generated control software also reflect the selected products, production processes, and production resources. For instance, Fig. 9 shows a generated control software based on selecting production processes, e.g., the *PopulatedPipe*, *InsertFork5*, and *InsertFork4*, and production resources, e.g., *LF_4_1* and *LF_4_2*, of the shift fork case study.

## 7. Discussion

This section concludes on the evaluation questions from the previous section and answers the research question raised in the introduction

```
Attribute "deltaFile": {
  description: "delta file for V4rdiac configuration",
  defaultValue: "", type: "String"
}

Process "WeldLock1": { name: "WeldLock1",
  implements: [ "WeldLock" ], inputs: [ "Lock1" ],
  deltaFile: "!DLock1"
}

Resource "WeldingRobot": { name: "WeldingRobot",
  isAbstract: true }
Resource "LaserWeldingRobot_01":{ name: "LaserWeldingRobot_01",
  implements: [ "LaserWeldingRobots" ],
  deltaFile: "DLaserWeldingRobot01"
}
```

Listing 4: Excerpt of the PPR–DSL model of the *shift fork* case study with the deltaFile attribute.

(cf. Section 1). Furthermore, it discusses the limitations of the prototype and threats to validity.

### 7.1. Observations and lessons learned

This section discusses the observations and lessons learned from the evaluation of the eIPSE approach.

*General comments.* To examine the applicability of eIPSE, we conducted a feasibility study with users experienced in the eIPSE approach on four published case studies (Meixner et al., 2021a) and an observational user study with users inexperienced in the eIPSE approach on a novel case study. We also investigated the reduction of the Process DM's configuration space by eIPSE and experimented with generating control code from such eIPSE configurations. These studies gave us valuable insight into potential benefits and perceived limitations of the eIPSE approach.

To this end, we go beyond our previous work (Meixner et al., 2022), i.e., by providing feedback on the eIPSE approach of users from different domains. This feedback indicates that the users perceived the eIPSE approach and toolchain as useful. In particular, users perceived the *externalization of knowledge* (Meixner, 2020) as valuable (requirements R1, R2, R4), the *separation of concerns for the configuration steps* (Ananieva et al., 2016; Fadhlillah et al., 2022b) as helpful (R3, R5), the *production process sequence exploration* as significant (Fang et al., 2013) (R1), and the *integration of variability models* as beneficial (Krüger et al., 2017) (R1, R2, R5). Furthermore, the users provided valuable feedback for approach improvements and future work.

The following paragraphs summarize the observations and lessons learned for each evaluation question.

*EQ1a application of eIPSE to case studies.* The primary lesson learned from applying eIPSE to the case studies was that modeling the PPR model combined with Steps 2 to 7 led to critical feedback. Additionally, we experienced that this feedback gained importance with the growing complexity of the product line of products and processes. For instance, while the *truck* case study contained no defects in the PPR model, it required significant improvement for the *rocker switch* and *water filter* case studies. This improvement mainly concerned missing requires or excludes constraints or additional CDC rules to limit the configuration options of the manufactured products suitably. Therefore, it is also possible to configure "semantically" incorrect products in the Product FM, which can be prevented by more carefully defining the variability in the PPR–DSL. This confirms research and industry voices (Feichtinger et al., 2022b) and further stakeholders from industry regarding the importance of these constraints (Nyberg, 2021). To this end, the eIPSE approach helped to reveal flaws, even in the already published PPR models (Meixner et al., 2021a). For instance, in Product FMs resulting from such PPR models, relevant products could not be configured anymore, the feature groups were incorrect, or the

Product FM allowed many more configurations than products in the product line. This also implies that software and CPPS product lines significantly differ regarding the configured output products, where, in the latter, each manufactured product must contribute to the cost and "return on investment" of the planned CPPS. Therefore, it requires support for variability model analysis to constrain the models and ensure that unintended products cannot be configured.

Second, the PPR–DSL (Meixner et al., 2021b), TRAVART (Feichtinger et al., 2021), FeatureIDE (Meinicke et al., 2017) FMs, DOPLER (Dhungana et al., 2011) DMs, and V4rdiac (Fadhlillah et al., 2022b) were originally designed as independent tools. Thus, we had to align the constraint formats and, respectively, the transformations.

*EQ1b application by engineers.* The first insight from observing engineers applying the eIPSE approach is that the engineering phase for the CPPS takes significantly longer (avg. 77 m) than the configuration phase (avg. 11 m). This is partly expected because application engineering is supposed to be faster than domain engineering. However, the editor support for defining the PPR–DSL may contribute to the time spent on the definition and update tasks and may need improvement. Still, the numbers confirm that much (mental) effort is invested in the product line definition. In more detail, we observed that most of the time was spent defining the products and the atomic production process steps rather than the production resources. The numbers also show that the configuration of the production process sequence in the Process DM took longer than the feature models' configuration. This indicates that creating a meaningful production process sequence is more complex but also requires means to evaluate and simulate the sequences economically. Furthermore, it implies that the time spent on different tasks requires a *separation of concerns in multidisciplinary engineering* (requirement R3, R5) and the *externalization of knowledge* (R1, R2, R4), which the engineers confirmed.

Secondly, the results show that the automatic creation and reduction of the variability model configuration space is fast and leads to a low time expenditure for the configuration of the models. This beats the definition of an additional product and potential production process steps as an increment in the PPR–DSL, which is still done manually with our approach. We argue that the eIPSE approach thus supports the challenge of an evolutionary creation of the product and its production process line.

Thirdly, the feedback of the subjects confirmed the usefulness of the eIPSE approach, in general. While the tool support for defining the PPR–DSL and the integration and feedback of the entire toolchain should be improved, the syntax seems to be straightforward for members of the intended user group despite a steep learning curve. Furthermore, the variability models could be configured quickly and intuitively because they only required little guidance from the supervising authors. However, the feedback explicitly points out *low code approaches* as a potential aim for industry.

*EQ2- Configuration space reduction.* Our results confirm our previous hypothesis (Meixner et al., 2022) and indicate that reducing the configuration space for decision models improves the guidance for engineers during production process exploration for valid and feasible production process sequences (requirement R1). We also argue that the eIPSE approach provides better reproducibility of the CPPS configuration through logging the selection sequence in the Process DM configuration. Nevertheless, it requires additional effort to explicitly model more domain-specific knowledge, such as throughput, cost, or risk, *to evaluate and simulate particular process sequences*.

*EQ3 - Generation of control artifacts.* The primary lesson learned was that the generated function block networks already present a working version of the control software for the configured CPPS. This shows that the integration of the variability models and their configuration works to a large extent (requirements R2, R4). Nevertheless, the control software engineers still need to connect some process blocks manually. For instance, the control software engineers must adjust the connections represented in the generated IEC 61499 function block networks to follow the process configuration.

Secondly, the control software engineers need to manually connect the generated production resource instances to the particular processes that use them. We could improve that by further incorporating the Process DM configuration as a basis for automatically generating the Delta models.

*Final remarks.* The evaluation with the (i) application by engineers to existing and a novel case study, (ii) an investigation of the Process DM configuration space, and (iii) the successful creation of control code artifacts shows that eIPSE is applicable to realistic cases and demands. However, the approach also implies an additional overhead. In particular, this concerns the explicit definition of the PPR model in the PPR–DSL, which also took the evaluation subjects the most time in the evaluation. The additional overhead also concerns the development of the toolchain and its future refinement. Nevertheless, we argue that the latter is a one-time effort while the former addresses the challenges of *implicit domain knowledge* and *configuration reproducibility* more than counterbalancing the additional effort. Still, an in-depth investigation of the additional effort implied by the eIPSE approach compared to completely manually finding feasible production process sequences needs to be conducted.

### 7.2. Answering the research questions

This section answers our research questions (cf. Section 1).

**RQ1** *How can CPPS engineers be supported in modeling, exploring, and configuring the combined variability of products, production processes, and production resources, to generate corresponding CPPS artifacts?*

To address this research question, this paper introduced the eIPSE approach. The approach aims at externalizing CPPS domain knowledge and the underlying variability by utilizing a domain-specific engineering artifact, i.e., the PPR–DSL, combined with two well-established variability models, i.e., FMs and DMs (cf. requirements **R1**, **R2**, and **R4** in Section 5.1). Furthermore, it aims to integrate the structural and behavioral variability of CPPS design aspects. It also provides defined feedback loops to proactively incorporate changes in requirements or design during CPPS engineering.

To this end, we go beyond the state-of-the-art (Ananieva et al., 2016; Fang et al., 2013; Krüger et al., 2019; Meixner, 2020; Meixner et al., 2019) by providing a framework for externalizing domain expert knowledge, integrating the structural and behavioral variability of CPPSs and their configuration, including the separation of concerns of different engineering domains.

**RQ2** *How and to what extent can CPPS design be automated using variability modeling and CPPS concepts?*

To address this research question, this work introduced the semi-automated eIPSE toolchain architecture supporting the modeling and configuration process of a CPPS' design based on a product configuration with a corresponding prototype. Therefore, we

- adapted the TRAVART transformation operations for production processes according to the new Process DM notation (cf. Fig. 4, Step 2b and **R2**),
- implemented TRAVART transformation operations for production resources (Step 2c and **R4**),
- implemented transformation operations for CDCs (Step 2d and **R5**),
- implemented the eIPSE DM Editor including the configuration space reduction of the Process DM (Steps 4 and 5), and
- adapted V4rdiac to integrate the transformed CDCs and included a pre-configuration step to read the configurations of the three variability models and generate CPPS implementation artifacts (Steps 6 to 8)

Thus, we go beyond the state-of-the-art by providing an integrated semi-automated toolchain for modeling and configuring the multidisciplinary structural and behavioral variability of CPPSs (Fadhlillah et al., 2022b; Meixner, 2020; Meixner et al., 2019). Additionally, we adapted the DOPLER DM (Dhungana et al., 2011) constraints so that they are solvable via standard SAT solvers, such as sat4j, and provided a DM editor not limited to the CPPS domain. To this end, we enable the transformation between the PPR model and state-of-the-art variability model types (Feichtinger and Rabiser, 2020a, 2021) integrating them into the *de facto* standard tool FeatureIDE (Meinicke et al., 2017) for further dissemination. Furthermore, we go beyond the state of the practice of the manual approach of engineering CPPS design also opening the way for better reuse of CPPS concepts.

Finally, we will discuss our work with industry partners to assess its practical impact. Therefore, we will investigate which concepts can already be implemented and what needs to be altered or adapted. Additionally, we will examine which parts of the prototype need to gain a higher technology readiness level for practical use. We argue that our approach can start a discussion on variability modeling and configuration in CPPS engineering and that the tools provide a solid foundation for future use.

### 7.3. Prototype limitations

The Process DM and eIPSE DM editor currently only support Boolean decisions rather than a broader range of decision types defined by the DOPLER approach (Dhungana et al., 2011). However, unlike the closed source DOPLER approach, our constraint definition syntax is SAT-solvable and thus usable with state-of-the-art software such as FeatureIDE. Integrating the multiple tools into the eIPSE toolchain still has room for improvement. For instance, aligning the syntax of the constraint definitions may be investigated.

### 7.4. Threats to validity

One threat to validity is that *several authors of the paper were involved in steps of the evaluation*. We tried to mitigate this threat by closely sticking to the provided engineering artifacts for the case studies and, where possible, gathering feedback from the engineers that provided the case studies. For the user study of the eIPSE toolchain, we involved five external subjects.

Due to its unstructured nature, another threat concerns the hard-to-measure *effort of the traditional manual approach*. The eIPSE approach mainly targets the automation of manual undocumented steps for CPPS engineering in alignment with the VDI 3695 (VDI, 2013).

Additionally, measuring the "time spent" for certain tasks might not be the best metric. However, in alignment with the VDI 3695 (VDI, 2013), we argue that the engineering time is a significant factor that should be elicited and optimized. Furthermore, it at least shows how fast relatively complex tasks can be completed with the eIPSE toolchain and certainly demonstrates that with the approach one would be faster than doing it manually. We try to demonstrate users' experiences by presenting their feedback.

Finally, the evaluation of *the approach's feasibility was conducted only for a limited set of case studies* of comparable size. Furthermore, the evaluation of *the approach's usefulness was conducted only for a single case study* and with *a small number of engineers*. Therefore, it is unclear how our approach would perform for systems of larger size and complexity. This may threaten the generalizability of the eIPSE approach. However, especially DMs in literature are smaller than the DMs created by our approach (Schmid et al., 2011). Consequently, our case studies, at least concerning the DMs, go beyond the state of the art.

## 8. Related work

This section presents approaches related to the eIPSE and work on variability modeling for CPPS.

Safdar et al. (2021) created a framework for supporting product configuration in the CPSs domain based on their evaluation of existing variability modeling approaches (Safdar et al., 2016). The framework mainly uses UML and OCL constraints for expressing CPS commonality and variability. The framework is also designed to support automated multi-stage and multi-level product configuration.

Fang (2019) developed a multi-view modeling approach for expressing variability in manufacturing. Variability is expressed in three different views: (1) software, (2) production process, and (3) plants' topology. The approach combines a feature meta-model with a topology and process meta-model to define a relation between variability from different views. Using this meta-model, one can create a topology and process models that are related to a FM when expressing variability in the manufacturing domain. At a later step, one can derive a customer-specific topology and process model that complies with the features selected from the FM.

Fadhlillah et al. (2022b) developed V4rdiac as a multidisciplinary variability management approach for CPPSs. V4rdiac is designed as a generic approach where CPPS engineers can use any types of variability model to express CPPS control software. CPPS engineers still need to decide which variability model best suits their domain. We use V4rdiac to showcase how our approach can be used to generate CPPS control software artifacts.

Existing works also use multiple FMs to model CPS system variability from different views or perspectives (Feldmann et al., 2015; Kowal et al., 2017; Rabiser et al., 2018; Cañete et al., 2022; Geraldi et al., 2020) in the machine manufacturing, industrial automation domain, and deployment of IoT application. They use cross-tree constraints or cross-model constraints to define the relation of features in the same or from different FMs. Expressing variability for industrial and complex software using multiple variability models is more beneficial in terms of maintainability and scalability compared to using a single variability model (Czarnecki et al., 2012; Kästner et al., 2012; Oliinyk et al., 2017). However, managing multi-view variability modeling, especially using heterogeneous types of variability models, is still a challenge (Krüger et al., 2017).

In contrast to existing works, our work expresses CPPS product, production process, and production resource variability. We use the PPR–DSL for expressing CPPS variability based on a modeling concept that CPPS engineers are already familiar with. The PPR–DSL offers a unified syntax for expressing product, production process, and production resource variability as well as dependencies among them. Additionally, we can transform our PPR–DSL into a Product FM, a Process DM, and a Resource FM to enable reasoning and configuration of a CPPS using existing product line tools (e.g., FeatureIDE). Additionally, production resources in our PPR–DSL can be related to CPPS artifacts (e.g., IEC 61499 control software). Thus, we provide a product configuration mechanism where we can generate customer-specific variants that conform to the selected product, production process, and production resource variants.

## 9. Conclusion and outlook

This paper introduced the eIPSE approach to decrease the manual and unstructured efforts while facilitating reproducibility in CPPS engineering. On top of our previous work (Feichtinger et al., 2020, 2021; Meixner et al., 2022, 2021b), we contributed (i) the eIPSE approach with additional steps to transform and configure production resource definitions and control software artifact generation, (ii) an extended prototype which realized the eIPSE approach (including a novel DM editor and configuration of SAT solvable DMs). We provide the corresponding artifacts in additional online material[1] and a demonstration video.[6] Furthermore, to investigate how the approach performs in practical settings, we: (i) conducted an evaluation of the feasibility in four published (ii)cpps case studies (Meixner et al., 2021a), (iii) conducted an observational user study with users inexperienced in the eIPSE approach and a novel case study, (iv) investigated the reduction of the CPPS configuration space, and (v) examined the generation of IEC 61499 (IEC, TC65/WG6, 2012) control software artifacts.

In this way, we go beyond the state-of-the-art (Fadhlillah et al., 2022b; Krüger et al., 2019; Meixner, 2020; Meixner et al., 2019) by providing a framework and semi-automated toolchain for CPPS variability modeling. This framework allows CPPS engineers to externalize better their domain expert knowledge, which comes primarily from experience and undocumented dependencies. Furthermore, the approach enables engineers to model and configure the multidisciplinary structural and behavioral variability of CPPSs while separating the concerns of the different engineering disciplines. Beyond that, the production process sequence exploration fosters reproducibility by recording the exploration steps in the toolchain. To evaluate the eIPSE approach, we collected the first feedback on the eIPSE approach from users from different domains who perceived the approach and toolchain as useful and recommended improvements for future work.

In future work, we aim to broaden the approach's applicability and perform further evaluation, initiating the next iteration cycle of Design Science. On the one hand, currently, the prototypes only support Boolean decisions, which may limit their usability in large industrial settings. When integrating advanced solvers, like SMTs, we plan also to support Non-Boolean decisions. On the other hand, the PPR–DSL may be improved in terms of editor support and by decoupling the processes from the production resources. Describing the overall CPPS variability may involve heterogeneous multi-view variability models for expressing the variability of different organizational units (e.g., business department, electrical engineering, or signal engineering). Thus, we also plan to extend further our eIPSE tool for creating a product configuration tool capable of enacting configuration options from heterogeneous multi-view variability models. Given this setup, we plan to conduct a large-scale evaluation with external practitioners from the CPPS domain to examine the feasibility of the eIPSE approach.

**CRediT authorship contribution statement**

**Kristof Meixner:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Kevin Feichtinger:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Hafiyyan Sayyid Fadhlillah:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Sandra Greiner:** Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft, Validation, Writing – original draft, Writing – review & editing. **Hannes Marcher:** Software, Writing – original draft. **Rick Rabiser:** Conceptualization, Funding acquisition, Methodology, Resources, Writing – original draft, Writing – review & editing. **Stefan Biffl:** Conceptualization, Funding acquisition, Resources, Writing – original draft.

**Declaration of competing interest**

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Data availability**

https://github.com/tuw-qse/eipse.

**Acknowledgments**

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. We explicitly want to thank our industry partners for their continuous support. Partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – CRC 1608 – 501798263. The water filter was invented, designed, and produced by Askwar Hilonga at Gongalimodel in Tanzania as a low cost frugal product line for cleaner water (UN SDG6). We thank Askwar Hilonga for providing and sharing this case study and Yazgül Fidan who initially conducted and analyzed the interviews with Askwar Hilonga. The truck models and their parts were designed, rendered, and 3D-printed at Czech Technical University in Prague - CIIRC. We thank Václav Jirkovský and Petr Novák for providing and sharing this case study. The chess piece case study was provided by TU Wien Pilotfabrik and the Center for Digital Production. We thank Alexander Raschendorfer and Sebastian Kropatschek for providing and sharing this case study. The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme.

## Appendix. Chess piece user study

This section describes the user study for evaluating the eIPSE approach.

### *A.1. Introduction*

We report the user study of the evaluation of the eIPSE toolchain. We investigate the modeling of an industrial product line and the subsequent production process exploration and production resource artifact derivation for a real-world *chess piece* product line designed at TU Wien Pilotfabrik.[17] Users assessed should go through the eIPSE process and evaluate the feasibility and usability of the approach and the tools.

We undertook the evaluation reported here as part of a collaboration between different academic and industrial initiatives. The subjects conducting the user study are, on the one hand, computer scientists and, on the other hand, engineers from companies that design or operate CPPSs.

### *A.2. Rationale*

The user study of the eIPSE evaluation was carried out in the focus of this paper and as part of three of the authors' dissertation projects. The research scope is to investigate variability modeling for CPPSs engineering and the required transformation of industrial artifacts to well-established variability models. The research in this context focuses on the reproducible exploration of production process sequences based on an integrated variability modeling approach. This variability modeling approach uses different types of variability models, i.e., feature models and decision models, to separate the concerns of the different stakeholders. There is limited published research on adopting state-of-the-art variability modeling and configuration approaches in the CPPS engineering industry, and the user study sought to contribute to the body of research in this area.

[17] TU Wien Pilotfabrik. https://www.pilotfabrik.at/.

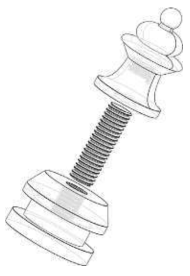


**Fig. 10.** CAD drawing of a pawn chess piece.

### A.3. Objective

The user study took place in an academic setting, which is also the primary audience for the user study. The overall objectives were as follows.

- To perform a eIPSE toolchain evaluation in a setting with subjects from different domains with the focus on CPPS engineering using a formal evaluation methodology.
- To learn from the evaluation about the following:
  - Can engineers model the functional view of CPPS for a small industrial product line with its products, process steps, and resources (domain engineering).
  - Can engineers efficiently explore and configure the design space for products, process sequences, and resources.
  - Does the integration of feature models and decision models allow for the configuration of a reasonable process sequence and corresponding resources based on a product configuration.
  - The time spent to model the functional view of CPPS for a small industrial product line (domain engineering).
  - The time spent to configure a CPPS design variant (application engineering).
  - The perceived usefulness of the eIPSE approach and toolchain for the engineers.
  - The lessons learned from using the eIPSE approach and toolchain for industrial product line modeling and configuration.
- To learn from the usage of eIPSE approach and toolchain.

These objectives are decidedly broad and ambitious. For conciseness, this chapter focuses on the objectives of learning from the evaluation about the time spent, the perceived usefulness, and the lessons learned.

### A.4. Chess piece use case

For the evaluation, we consider the *chess piece* use case from TU Pilotfabrik. The use case concerns a CPPS that manufactures the six chess piece types, i.e., king, queen, bishops, knights, rooks, and pawns. Fig. 10 shows a CAD drawing of the pawn chess piece.

Each chess piece consists of a *base*, a *body*, and a threaded *rod* that connects them.

The *base* is produced from *aluminum bars* of 1 m length on a *turning machine*. The *aluminum bar* is *loaded into* the *turning machine* with a *bar loader*. The *turning machine cuts* the *aluminum bar* into the raw *bases* of suitable length for further processing. These *bases*, which come in two variants, are *turned* on the *turning machine* to get their specific shape. The king and the queen have a base with *two circumferential reamings* that are *carved into the aluminum*. The other chess pieces have a base that has *one circumferential reaming*. After creation, a *laser profiler* measures the bases for turning accuracy.

The *body* of the particular chess pieces is *3D-printed* from Polyactic Acid (PLA) on an industrial *3D printer*.

The *base* and the *body* each have a hole with a thread carved respectively printed in the middle. The threads each have a diameter for a standardized M6 *rod*.[18] Similarly to the base, the threaded *rod* comes in two variants, one with *20 millimeters* and one with *30 millimeters* in length.

The *base*, the *body*, and the threaded *rod* are assembled in an *assembly station*, where each individual part needs to be loaded into the station. The parts need to be *screwed* together, which can be done in an arbitrary sequence.

### A.5. Subject guideline

Conduct the eIPSE process, as described in Section 5, for the chess piece use case for an imaginative CPPS.

*Chess piece product line modeling.* This task represents *Step 1* of the eIPSE approach in Fig. 4. As a user, create a functional PPR–DSL model of the chess piece product line in the Sublime text editor. Use the provided cheat sheet for the syntax of the PPR–DSL. The model shall represent three parts, which are (i) the partial and final products that should be manufactured by the CPPS, (ii) the atomic process steps that create the products with their required input products, predecessors, and resources, as well as (iii) the production resources that can execute a particular process required to manufacture a product.

**Products** For the chess piece use case, create `partial` products and `final` products in the PPR–DSL. Find out which partial products you could group using `abstract` parent products. Furthermore, define which of the partial products `exclude` each other because a similar partial product more or less supplements them.

**Processes** As a user, think about how to assemble partial products via specific atomic "manufacturing" processes. Realize these atomic process steps in the PPR–DSL and, similar to the products, group them on their `abstract` products and `exclude` process steps that you deem unnecessary. Furthermore, consider which process steps in a particular assembly process need to be direct predecessors and refer to them as needed in the required section.

**Resources** As a user, model the resources similar to products. For modeling them, you should apply similar rules as for the product variability model.

*Model transformation.* This task represents *Step 2* of the eIPSE approach in Fig. 4. Use TRAVART from the IPSE toolchain from the command line to transform the chess piece PPR–DSL model to the product and resource feature model (`uvl` file extension) and the process decision model (`dmodel` file extension).

*Iterative process exploration.* This task represents *Steps 3 to 5* of the eIPSE approach in Fig. 4. Configure a desired product of the chess piece product line, i.e., one of the six chess piece types, using the configurator for feature models in Eclipse (Step 3 in Fig. 4) by ticking the checkboxes for the features. Then, generate a reduced *decision model configuration* (`dconfig` file extension) by right-clicking on the decision model and selecting the configuration file of the previously configured product (`xml` file ending) (Step 4 in Fig. 4). Open the decision model configuration file in the *decision model configurator* and explore feasible production process sequences for the configured chess piece by ticking the decision checkboxes (Step 5 in Fig. 4). Therefore, you can investigate the process sequence in the right pane of the decision model configurator.

[18] ISO metric screw threads: https://w.wiki/_wm23.

*Resource configuration and artifact generation.* This task represents *Steps 6 to 8* of the eIPSE approach in Fig. 4. Use the delta models (delta file extension) that are prepared according to the features or decisions in product *feature model*, process *decision model*, and resource *feature model* that might affect the control source code of the CPPS. Additionally, use the prepared delta configuration file (deltaconf file extension) in V4rdiac to map the delta models into its corresponding feature or decision. Then, use V4rdiac to load the previously configured product and production processes sequence and configure the desired resources. You can generate the control source code for the CPPS according to the selected features or decisions after the configuration is finished.

### A.6. Study protocol

No formal study protocol was developed or maintained for the user study.

### A.7. Evaluation questions

We formulated the following questions for the evaluation.

- Is it feasible to apply the eIPSE approach?
  - in different real-world case studies?
  - by engineers from heterogeneous backgrounds?

### A.8. Methods of data collection

We used *shadowing* (Singer et al., 2008) to investigate 5 subjects during performing the eIPSE approach on the *chess piece* use case. Due to the distributed locations of the subjects performing the evaluation, we used Zoom to connect them to the eIPSE toolchain and provided them with remote control. At least two of the authors shadowed the subjects during the evaluation sessions. One author helped the evaluation subject if questions arose, additional explanations were required, or the subjects were stuck in the process. The other authors present took notes for later investigation. Furthermore, one author stopped the time for each of the activities and steps of the evaluation process. Additionally, we recorded the evaluation sessions to replay them later during the internal result analysis. Afterward, we asked the subjects about the perceived usefulness of the toolchain and the lessons learned during the process.

### A.9. Methods of data analysis

No particular strategy for coding the notes taken was used. We extracted quotes from the notes that concerns the perceived usefulness and could lead to improvements of the approach and the toolchain.

### A.10. Case selection strategy

The case itself was selected on the basis that its main feature, i.e., the chess pieces, are well known by most people. Furthermore, the production of the chess pieces seems lucid enough for engineers of different domains to be manageable. To this end, it should be straightforward to understand how the possible production process could be modeled. Yet, the case appears to be complex enough to properly investigate the problem of the large configuration space.

### A.11. Data selection strategy

The strategy for selecting data was driven primarily by the activities defined in the subject guideline.

### A.12. Replication strategy

There was no strategy for replication on the basis that there was no comparable evaluation previously conducted or even comparable evaluations of other technologies. We aim that future case studies of the evaluation of the eIPSE approach adopt the here described description for partial replication.

### A.13. Quality assurance

To help ensure that data collected were representative of a broad range of stakeholders in the domain of CPPS engineering, we selected engineers and stakeholders from different companies and domains.

### A.14. Data collection

We used *shadowing* (Singer et al., 2008) to investigate 5 subjects during performing the eIPSE approach on the *chess piece* use case. Due to the distributed locations of the subjects performing the evaluation, we used Zoom to connect them to the eIPSE toolchain and provided them with remote control. At least two of the authors shadowed the subjects during the evaluation sessions. One author helped the evaluation subject if questions arose, additional explanations were required, or the subjects were stuck in the process. The other authors present took notes for later investigation. Furthermore, one author stopped the time for each of the activities and steps of the evaluation process. Additionally, we recorded the evaluation sessions to replay them later during the internal result analysis. Afterward, we asked the subjects about the perceived usefulness of the toolchain and the lessons learned during the process.

## References

Ananieva, S., Kowal, M., Thüm, T., Schaefer, I., 2016. Implicit constraints in partial feature models. In: 7th International FOSD Workshop. FOSD@SPLASH 2016, pp. 18–27.
Apel, S., Batory, D., Kästner, C., Saake, G., 2013. Feature-Oriented Software Development: Concepts and Implementation. Springer.
Biffl, S., Gerhard, D., Lüder, A., 2017. Introduction to the multi-disciplinary engineering for cyber-physical production systems. In: Multi-Disciplinary Engineering for Cyber-Physical Production Systems. Springer, pp. 1–24.
Campbell, G.H., Faulk, S.R., Weiss, D.M., 1990. Introduction To Synthesis. Technical Report, INTRO_SYNTHESIS_PROCESS-90019-N, Software Productivity Consortium, Herndon, VA, USA.
Cañete, A., Amor, M., Fuentes, L., 2022. Supporting IoT applications deployment on edge-based infrastructures using multi-layer feature models. J. Syst. Softw. 183, 111086.
Clarke, D., Helvensteijn, M., Schaefer, I., 2015. Abstract delta modelling. Math. Struct. Comput. Sci. 25 (3), 482–527.
Clements, P., Northrop, L., 2002. Software Product Lines. Addison-Wesley Boston.
Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., Wąsowski, A., 2012. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. In: 6th International Workshop on Variability Modeling of Software-Intensive Systems. VaMoS '12, ACM, pp. 173–182.
Dhungana, D., Grünbacher, P., Rabiser, R., 2011. The DOPLER Meta-Tool for Decision-Oriented Variability Modeling: A Multiple Case Study. Automat. Softw. Eng. 18 (1), 77–114.
Drath, R., 2009. Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA. Springer.
Drath, R., 2021. Automationml - a practical guide. De Gruyter Oldenbourg,
Drath, R., Luder, A., Peschke, J., Hundt, L., 2008. Automationml - the glue for seamless automation engineering. In: 13th International Conference on Emerging Technologies and Factory Automation. ETFA '08, IEEE, pp. 616–623.
Fadhlillah, H.S., Feichtinger, K., Bauer, P., Kutsia, E., Rabiser, R., 2022a. V4rdiac: tooling for multidisciplinary variability management in cyber-physical production systems. In: 26th ACM International Systems and Software Product Line Conference - Volume B. SPLC '22, ACM, pp. 34–37.
Fadhlillah, H.S., Feichtinger, K., Meixner, K., Sonnleithner, L., Rabiser, R., Zoitl, A., 2022b. Towards Multidisciplinary Delta-oriented variability management in cyber-physical production systems. In: 16th International Working Conference on Variability Modelling of Software-Intensive Systems. VaMoS '22, ACM, pp. 13:1–13:10.

Fadhlillah, H.S., Fernández, A.M.G., Rabiser, R., Zoitl, A., 2023a. Managing cyber-physical production systems variability using V4rdiac: Industrial experiences. In: 27th ACM International Systems and Software Product Line Conference - Volume A. SPLC '23, ACM, pp. 223–233.

Fadhlillah, H.S., Sharma, S., Gutierrez Fernandez, A.M., Rabiser, R., Zoitl, A., 2023b. Delta modeling in IEC 61499: Expressing control software variability in cyber-physical production systems. In: 28th International Conference on Emerging Technologies and Factory Automation. ETFA '23, IEEE, pp. 1–8.

Fang, M., 2019. Model-Based Software Derivation for Industrial Automation Management Systems (Ph.D. thesis). Technische Universität Kaiserslautern.

Fang, M., Leyh, G., Elsner, C., Dörr, J., 2013. Challenges in managing behavior variability of production control software. In: PLEASE@ICSE. IEEE Computer Society, pp. 21–24.

Feichtinger, K., Meixner, K., Biffl, S., Rabiser, R., 2022a. Evolution support for custom variability artifacts using feature models: A study in the cyber-physical production systems domain. In: 20th International Conference on Software and Systems Reuse. Springer, pp. 79–84.

Feichtinger, K., Meixner, K., Rabiser, R., Biffl, S., 2020. Variability Transformation from Industrial Engineering Artifacts: An Example in the Cyber-Physical Production Systems Domain. In: 24th ACM International Systems and Software Product Line Conference - Volume B. SPLC '20, ACM, pp. 65–73.

Feichtinger, K., Meixner, K., Rinker, F., Koren, I., Eichelberger, H., Heinemann, T., Holtmann, J., Konersmann, M., Michael, J., Neumann, E.-M., Pfeiffer, J., Rabiser, R., Riebisch, M., Schmid, K., 2022b. Industry voices on software engineering challenges in cyber-physical production systems engineering. In: 27th International Conference on Emerging Technologies and Factory Automation. ETFA '22, IEEE, pp. 1–8.

Feichtinger, K., Rabiser, R., 2020a. Towards transforming variability models: Usage scenarios, required capabilities and challenges. In: 24th ACM International Systems and Software Product Line Conference - Volume B. SPLC '20, ACM, pp. 44–51.

Feichtinger, K., Rabiser, R., 2020b. Variability model transformations: Towards unifying variability modeling. In: 46th Euromicro Conference on Software Engineering and Advanced Applications. IEEE.

Feichtinger, K., Rabiser, R., 2021. How flexible must a transformation approach for variability models and custom variability representations be? In: 24th ACM International Systems and Software Product Line Conference - Volume B. SPLC '21, ACM, pp. 69–72.

Feichtinger, K., Stöbich, J., Romano, D., Rabiser, R., 2021. TRAVART: An approach for transforming variability models. In: 15th International Working Conference on Variability Modelling of Software-Intensive Systems. VaMoS '21, ACM, pp. 8:1–8:10.

Feldmann, S., Legat, C., Vogel-Heuser, B., 2015. Engineering support in the machine manufacturing domain through interdisciplinary product lines: An applicability analysis. IFAC-PapersOnLine 28 (3), 211–218.

Galster, M., Weyns, D., Tofan, D., Michalik, B., Avgeriou, P., 2013. Variability in software systems-a systematic literature review. IEEE Trans. Softw. Eng. 40 (3), 282–306.

Geraldi, R.T., Reinehr, S., Malucelli, A., 2020. Software product line applied to the internet of things: A systematic literature review. Inf. Softw. Technol. 124, 106293.

Gunes, V., Peter, S., Givargis, T., Vahid, F., 2014. A survey on concepts, applications, and challenges in cyber-physical systems.. KSII Trans. Internet Inf. Syst. 8 (12).

Hevner, A.R., 2007. A three cycle view of design science research. Scand. J. Inf. Syst. 19 (2), 4.

Hevner, A.R., March, S.T., Park, J., Ram, S., 2008. Design science in information systems research. MIS Q. 28 (1), 6.

Hubaux, A., Xiong, Y., Czarnecki, K., 2012. A user survey of configuration challenges in linux and ecos. In: 6th International Workshop on Variability Modeling of Software-Intensive Systems. VaMoS '12, ACM, pp. 149–155.

IEC, 2014. IEC 62714-1, Engineering data exchange format for use in industrial automation systems engineering - Automation markup language - Part 1: Architecture and general requirements.

IEC, TC65/WG6, 2012. IEC 61499-1, Function Blocks - part 1: Architecture: Edition 2.0.

Järvenpää, E., Siltala, N., Hylli, O., Lanz, M., 2019. Implementation of capability matchmaking software facilitating faster production system design and reconfiguration planning. J. Manuf. Syst. 53, 261–270.

Jazdi, N., Maga, C., Göhner, P., Ehben, T., Tetzner, T., Löwen, U., 2010. Mehr systematik für den anlagenbau und das industrielle lösungsgeschäft - gesteigerte effizienz durch domain engineering, 58 (9). pp. 524–532.

Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S., 1990. Feature-oriented domain analysis (FODA) feasibility study. Technical Report, Carnegie-Mellon Univ., Pittsburgh, Pa, Software Engineering Inst..

Kästner, C., Ostermann, K., Erdweg, S., 2012. A variability-aware module system. In: ACM International Conference on Object Oriented Programming Systems Languages and Applications. OOPSLA '12, ACM, pp. 773–792.

Kowal, M., Ananieva, S., Thüm, T., Schaefer, I., 2017. Supporting the Development of Interdisciplinary Product Lines in the Manufacturing Domain. IFAC-PapersOnLine 50 (1), 4336–4341.

Krüger, J., Mukelabai, M., Gu, W., Shen, H., Hebig, R., Berger, T., 2019. Where is my feature and what is it about? A case study on recovering feature facets. J. Syst. Softw. 152, 239–253.

Krüger, J., Nielebock, S., Krieter, S., Diedrich, C., Leich, T., Saake, G., Zug, S., Ortmeier, F., 2017. Beyond software product lines: Variability modeling in cyber-physical systems. In: 21st International Systems and Software Product Line Conference - Volume A. SPLC '17, ACM, pp. 237–241.

Lee, S., 1989. Disassembly planning based on subassembly extraction. In: Third ORSA/TIMS Conference on Flexible Manufacturing System. pp. 383–388.

Meinicke, J., Thüm, T., Schröter, R., Benduhn, F., Leich, T., Saake, G., 2017. Mastering Software Variability with FeatureIDE. Springer.

Meixner, K., 2020. Integrating Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering. In: 24th ACM International Systems and Software Product Line Conference - Volume B. SPLC '20, ACM, pp. 96–103.

Meixner, K., Feichtinger, K., Rabiser, R., Biffl, S., 2021a. A reusable set of real-world product line case studies for comparing variability models in research and practice. In: 25th International Systems and Software Product Line Conference - Volume B. SPLC '21, ACM, pp. 105–112.

Meixner, K., Feichtinger, K., Rabiser, R., Biffl, S., 2022. Efficient Production Process Variability Exploration. In: 16th International Working Conference on Variability Modelling of Software-Intensive Systems. VaMoS '22, ACM, pp. 14:1–14:9.

Meixner, K., Rabiser, R., Biffl, S., 2019. Towards modeling variability of products, processes and resources in cyber-physical production systems engineering. In: 23rd International Systems and Software Product Line Conference - Volume B. SPLC '19, ACM, pp. 68:1–68:8.

Meixner, K., Rabiser, R., Biffl, S., 2020. Feature identification for engineering model variants in cyber-physical production systems engineering. In: 14th International Working Conference on Variability Modelling of Software-Intensive Systems. VaMoS '20, ACM, pp. 18:1–18:5.

Meixner, K., Rinker, F., Marcher, H., Decker, J., Biffl, S., 2021b. A domain-specific language for product-process-resource modeling. In: 26th International Conference on Emerging Technologies and Factory Automation. ETFA '21, IEEE, pp. 1–8.

Monostori, L., 2014. Cyber-physical production systems: Roots, expectations and R&D challenges. Procedia CIRP 17, 9–13.

Nyberg, M., 2021. Generating safety cases for large-scale industrial product lines. URL https://splc2021.net/program/keynotes, Keynote at 25th ACM International Systems and Software Product Line Conference.

Oliinyk, O., Petersen, K., Schoelzke, M., Becker, M., Schneickert, S., 2017. Structuring automotive product lines and feature models: an exploratory study at opel. Requir. Eng. 22 (1), 105–135.

Paetzold, K., 2017. Product and systems engineering/ca* tool chains. In: Multi-Disciplinary Engineering for Cyber-Physical Production Systems. Springer, pp. 27–62.

Rabiser, D., Prähofer, H., Grünbacher, P., Petruzelka, M., Eder, K., Angerer, F., Kromoser, M., Grimmer, A., 2018. Multi-purpose, multi-level feature modeling of large-scale industrial software systems. Softw. Syst. Model. 17 (3), 913–938.

Runeson, P., Host, M., Rainer, A., Regnell, B., 2012. Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons.

Safdar, S.A., Lu, H., Yue, T., Ali, S., Nie, K., 2021. A framework for automated multi-stage and multi-step product configuration of cyber-physical systems. In: Software and Systems Modeling. 20 (1), 211–265.

Safdar, S.A., Yue, T., Ali, S., Lu, H., 2016. Evaluating variability modeling techniques for supporting cyber-physical system product line engineering. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9959 LNCS (1).1–19,

Schäfer, A., Becker, M., Andres, M., Kistenfeger, T., Rohlf, F., 2021. Variability realization in model-based system engineering using software product line techniques: An industrial perspective. In: 25th ACM International Systems and Software Product Line Conference - Volume A. SPLC '21, ACM, pp. 25–34.

Schleipen, M., Lüder, A., Sauer, O., Flatt, H., Jasperneite, J., 2015. Requirements and concept for plug-and-work. at-Automatisierungstechnik 63 (10), 801–820.

Schmid, K., John, I., 2004. A customizable approach to full lifecycle variability management. Sci. Comput. Program. 53 (3), 259–284.

Schmid, K., Rabiser, R., Grünbacher, P., 2011. A comparison of decision modeling approaches in product lines. In: 5th International Workshop on Variability Modelling of Software-Intensive Systems. VaMoS '11, ACM, pp. 119–126.

Shull, F., Singer, J., Sjøberg, D.I., 2007. Guide To Advanced Empirical Software Engineering. Springer.

Singer, J., Sim, S.E., Lethbridge, T.C., 2008. Software engineering data collection for field studies. Guide to advanced empirical software engineering 9–34.

Sundermann, C., Feichtinger, K., Engelhardt, D., Rabiser, R., Thüm, T., 2021. Yet another textual variability language? A community effort towards a unified language. In: 25th ACM International Systems and Software Product Line Conference - Volume A. SPLC '21, ACM, pp. 136–147.

Tolio, T., Ceglarek, D., ElMaraghy, H.A., Fischer, A., Hu, S.J., Laperrière, L., Newman, S.T., Váncza, J., 2010. SPECIES—Co-evolution of products, processes and production systems. CIRP Annals 59 (2), 672–693.

VDI, 2005. VDI/VDE 3682: Formalised process descriptions. Beuth Verlag.

VDI, 2013. VDI/VDE 3695: Engineering of industrial plants.. Beuth Verlag, 5 parts.

Wieringa, R.J., 2014. Design Science Methodology for Information Systems and Software Engineering. Springer.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in software engineering. Springer Science & Business Media.
Zhang, B., Duszynski, S., Becker, M., 2016. Variability mechanisms and lessons learned in practice. In: 1st International Workshop on Variability and Complexity in Software Design. VACE '16, Association for Computing Machinery, New York, NY, USA, pp. 14–20.
Zoitl, A., Strasser, T., Valentini, A., 2010. Open source initiatives as basis for the establishment of new technologies in industrial automation: 4DIAC a case study. In: 2010 IEEE International Symposium on Industrial Electronics. IEEE, pp. 3817–3819.

**Kristof Meixner** is a researcher and Ph.D. student at the Christian Doppler Laboratory SQI at TU Wien, Austria. He completed his master's degree in 2018 in Business Informatics at TU Wien. His research interests include empirical software engineering, model-based software engineering, software quality, and systems and software product line engineering for Cyber–Physical Production Systems. He regularly co-organizes several community workshops at international conferences.

**Kevin Feichtinger** is a Postdoctoral Researcher at the Dependability of Software-intensive Systems Group at Karlsruhe Institute of Technology, Germany. He completed his Ph.D. studies in computer science major software engineering at the LIT Cyber–Physical Systems Lab at Johannes Kepler University Linz in 2023. His current research interests include software product lines, variability modeling, model transformations, software evolution, software consistency, and software development for Cyber–Physical Systems (CPS).

**Hafiyyan Sayyid Fadhlillah** is a Ph.D. student at the CDL VaSiCS, LIT CPS Lab at the Johannes Kepler University Linz, Austria (JKU). He completed his master's degree in 2017 in Computer Science at Universitas Indonesia. His current research interests include variability modeling, variability mechanism, software configuration, Software Development for Cyber–Physical (Production) Systems, and Web Application Development.

**Sandra Greiner** is postdoctoral researcher at the University of Bern. She earned her Ph.D. in Software Engineering at the University of Bayreuth and studied Computer Science (M.Sc.) and its application with Engineering Sciences (B.Sc.). During a research visit at the IT University of Copenhagen, granted as a scholarship by the German Academic Exchange Service (DAAD), she investigated synchronizing evolving variability information in software artifacts. Her research interests include model-driven software-engineering, software product line engineering, and adopting software engineering to enhance industrial applications. She regularly co-organizes several community workshops and is particularly interested in systematically managing software variability over time.

**Hannes Marcher** is a student assistant at the Christian Doppler Laboratory SQI at TU Wien. His research interests include aspects of reuse and variability in Cyber–Physical Production Systems and data science.

**Rick Rabiser** is a full university professor for Software Engineering in Cyber–Physical Systems and head of the LIT Cyber–Physical Systems Lab as well as the Christian Doppler Laboratory for Mastering Variability in Software-intensive Cyber-Physical Production Systems at Johannes Kepler University Linz, Austria. Together with his team, he performs research on variability management, systems and software product lines, software evolution, automated software engineering, and usability of software engineering tools.

**Stefan Biffl** is a professor of Software Engineering at TU Wien, the TUW module lead in the Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle, and a supervisor in the TUW Doctoral College Trustworthy Autonomous Cyber–Physical Systems. His research interests include Software and Systems Engineering Process Improvement, Multi–view Model Quality Analysis and Improvement, Software Architecture, in particular, Collective Intelligence Systems, and Performance–Driven Knowledge Engineering for Software–Intensive Systems.

## 5.4 Additional Publications

| Ref# | Title | Venue | Year | Author |
|---|---|---|---|---|
| [206] | “Towards Flexible and Automated Testing in Production Systems Engineering Projects” | ETFA | 2018 | 1st |
| [119] | “Towards Combined Process and Tool Variability Management in Software Testing” | VaMoS | 2019 | 1st |
| [121] | “Towards Model-driven Verification of Robot Control Code using Abstract Syntax Trees in Production Systems Engineering” | MODELSWARD | 2019 | 1st |
| [88] | “Production-aware analysis of multi-disciplinary systems engineering processes” | ICEIS | 2019 | 3rd |
| [11] | “Software Engineering Risks from Technical Debt in the Representation of Product/ion Knowledge” | SEKE | 2019 | 4th |
| [92] | “Product/ion-Aware Modeling Approaches that Support Tracing Design Decisions” | INDIN | 2019 | 2nd |
| [34] | “Securing the testing process for industrial automation software” | Computers and Security | 2019 | 2nd |
| [120] | “Supporting Domain Experts by using Model-Based Equivalence Class Partitioning for Efficient Test Data Generation” | ETFA | 2019 | 1st |
| [89] | “Efficient Production System Resource Exploration Considering Product/ion Requirements” | ETFA | 2019 | 2nd |
| [91] | “Extending the Formal Process Description towards Consistency in Product/ion-Aware Modeling” | ETFA | 2019 | 2nd |
| [90] | “A Meta-Model for Representing Consistency as Extension to the Formal Process Description” | ETFA | 2019 | 2nd |
| [146] | “Engineering Roles and Information Modeling for Industry 4.0 Production System Engineering” | ETFA | 2019 | nth |
| [163] | “Towards Support of Global Views on Common Concepts employing Local Views” | ETFA | 2019 | nth |
| [207] | “Towards a Hybrid Process Model Approach in Production Systems Engineering” | EuroSPI | 2019 | nth |
| [122] | “Investigating the Performance of selected Data Storage Concepts for AutomationML Models” | IECON | 2019 | 1st |
| [209] | “Efficient and Flexible Test Automation in Production Systems Engineering” | Security and Quality in CPS Engineering | 2019 | 2nd |
| [87] | “Product/ion-Aware Analysis of Collaborative Systems Engineering Processes” | Security and Quality in CPS Engineering | 2019 | 3rd |
| [208] | “Test Reporting at a Large-Scale Austrian Logistics Organization: Lessons Learned and Improvement” | PROFES | 2019 | 2nd |
| [126] | “Feature identification for engineering model variants in cyber-physical production systems engineering” | VaMoS | 2020 | 1st |
| [124] | “Efficient Test Case Generation from Product and Process Model Properties and Preconditions” | ETFA | 2020 | 1st |
| [123] | “Towards a Domain-Specific Language for Product-Process-Resource Constraints” | ETFA | 2020 | 1st |
| [125] | “Modeling Expert Knowledge for Optimal CPPS Resource Selection for a Product Portfolio” | ETFA | 2020 | 1st |

| | | | | |
|---|---|---|---|---|
| [210] | “Towards Model Consistency Representations in a Multi-Disciplinary Engineering Network” | ETFA | 2020 | 3rd |
| [53] | “Decision Support for Frugal Products and Production Systems based on Product-Process-Resource-Skill and Variability Models” | Procedia CIRP | 2021 | 3rd |
| [165] | “Continuous Integration in Multi-view Modeling: A Model Transformation Pipeline Architecture for Production Systems Engineering” | MODELSWARD | 2021 | 3rd |
| [12] | “Multi-view-Model Risk Assessment in Cyber-Physical Production Systems Engineering” | MODELSWARD | 2021 | 3rd |
| [38] | “Towards Multidisciplinary Delta-Oriented Variability Management in Cyber-Physical Production Systems” | VaMoS | 2022 | 3rd |
| [212] | “Industry 4.0 Asset-based Risk Mitigation for Production Operation” | CASE | 2021 | nth |
| [14] | “An Industry 4.0 Asset-Based Coordination Artifact for Production Systems Engineering” | CBI | 2021 | nth |
| [45] | “A Systematic Study as Foundation for a Variability Modeling Body of Knowledge” | SEAA | 2021 | 2nd |
| [60] | “Fourth International Workshop on Variability and Evolution of Software-Intensive Systems (VariVolution 2021)” | SPLC | 2021 | 3rd |
| [101] | “Towards the Representation of Cross-Domain Quality Knowledge for Efficient Data Analytics” | ETFA | 2021 | nth |
| [13] | “Towards Efficient Asset-Based Configuration Management with a PPR Asset Directory” | ETFA | 2021 | 2nd |
| [211] | “Product-Process-Resource Asset Networks as Foundation for Improving CPPS Engineering” | ETFA | 2021 | 3rd |
| [112] | “Modelling Engineered Object Dependencies in an AutomationML-based Tool Chain” | ETFA | 2021 | 2nd |
| [180] | “Reducing Risk in Industrial Bin Picking with PPRS Configuration and Dependency Management” | ETFA | 2021 | 2nd |
| [164] | “Towards Efficient Generation of a Multi-Domain Engineering Graph with Common Concepts” | ETFA | 2021 | 2nd |
| [161] | “Precisely and Persistently Identifying and Citing Arbitrary Subsets of Dynamic Data” | HDSR | 2021 | nth |
| [166] | “Efficient Multi-view Change Management in Agile Production Systems Engineering” | ICEIS | 2022 | nth |
| [47] | “Evolution Support for Custom Variability Artifacts Using Feature Models: A Study in the Cyber-Physical Production Systems Domain” | ICSR | 2022 | 2nd |
| [15] | “Risk-Driven Derivation of Operation Checklists from Multi-Disciplinary Engineering Knowledge” | INDIN | 2022 | nth |
| [167] | “Risk and Engineering Knowledge Integration in Cyber-physical Production Systems Engineering” | SEAA | 2022 | 2nd |
| [48] | “Industry Voices on Software Engineering Challenges in Cyber-Physical Production Systems Engineering” | ETFA | 2022 | 2nd |
| [76] | “Towards Design Patterns for Production Security” | ETFA | 2022 | 3rd |
| [134] | “A Coordination Artifact for Multi-disciplinary Reuse in Production Systems Engineering” | ETFA | 2022 | 1st |
| [58] | “Capabilities and Skills in Manufacturing: A Survey Over the Last Decade of ETFA” | ETFA | 2022 | 3rd |
| [102] | “Designing a Digital Shadow for Efficient, Low-Delay Analysis of Production Quality Risk” | ETFA | 2022 | nth |
| [16] | “Towards Coordinating Production Reconfiguration” | ETFA | 2022 | 2nd |

| | | | | |
|---|---|---|---|---|
| [62] | "Fifth International Workshop on Variability and Evolution of Software-Intensive Systems (VariVolution 2022)" | SPLC | 2022 | 2nd |
| [103] | "Analysis of Quality Issues in Production With Multi-view Coordination Assets" | IFAC MIM | 2022 | nth |
| [114] | "Engineering Data Treasures, Their Collection and Use" | IFAC MIM | 2022 | 2nd |
| [97] | "A reference model for common understanding of capabilities and skills in manufacturing" | AT | 2023 | nth |
| [170] | "Traceable Multi-view Model Integration: A Transformation Pipeline for Agile Production Systems Engineering" | SNCS | 2023 | 3rd |
| [99] | "Challenges and Opportunities of DevOps in Cyber-Physical Production Systems Engineering" | ICPS | 2023 | 3rd |
| [17] | "Validating Production Test Scenarios with Cyber-Physical System Design Models" | CBI | 2023 | nth |
| [168] | "Multi-view FMEA Re-validation: Efficient Risk and Engineering Knowledge Integration in Agile Production Systems Engineering" | MODELSWARD | 2023 | nth |
| [63] | "Sixth International Workshop on Variability and Evolution of Software-Intensive Systems (VariVolution 2023)" | SPLC | 2023 | 3rd |
| [98] | "Implementing DevOps Practices in CPPS Using Microservices and GitOps" | ETFA | 2023 | 3rd |
| [188] | "IEC 61499 Skill-based Distributed Design Pattern" | ETFA | 2023 | nth |
| [64] | "Maturity Evaluation of Domain-Specific Language Ecosystems for Cyber-Physical Production Systems" | ETFA | 2023 | nth |
| [18] | "Towards Test-Driven Performance Validation of a Flexible Cyber-Physical Production System" | ETFA | 2023 | 2nd |
| [169] | "Organizing Multi-Domain Change Impact Analysis in Cyber-Physical Production Systems Engineering" | ETFA | 2023 | 3rd |
| [113] | "Consistent Extension of Networks of Digital Representations of Production System Assets" | ETFA | 2023 | 3rd |
| [65] | "Dataset: Maturity Evaluation of Domain-Specific Language Ecosystems for Cyber-Physical Production Systems (Version 1)" | | 2023 | nth |
| [77] | "Interdisciplinary Production Risk Exploration: A Grounded Approach to Integrate Data- and Knowledge-Driven Analytics" | CIRP CMS | 2023 | nth |
| [66] | "Generative AI And Software Variability – A Research Vision" | VaMoS | 2024 | nth |
| [138] | "On Configuration Sequences in Feature Models" | VaMoS | 2024 | 1st |
| [173] | "Survey of Practitioner Needs and Approaches for Multi-Domain Change Management in Cyber-Physical Production Systems Engineering" | ETFA | 2024 | 3rd |
| [172] | "Multi-Domain Modeling for Change Management in Cyber-Physical Production Systems Engineering" | ETFA | 2024 | 2nd |
| [171] | "Graph-Based Change Impact Visualization for Agile Cyber-Physical Production Systems Engineering" | ETFA | 2024 | 2nd |
| [139] | "Rolling the Dice – Rethinking the RAMI 4.0 Perspectives" | ETFA | 2024 | 1st |
| [110] | "Identifying required Knowledge for Production System Digitalization projects" | ETFA | 2024 | 4th |
| [111] | "Representing Property Dependencies within AutomationML Based Digital Twins" | ETFA | 2024 | th |
| [136] | "Variability modeling of products, processes, and resources in cyber-physical production systems engineering" | SPLC | 2024 | 1st |

Table 5.1: Chronological list of additional publications not in focus of the cumulative thesis.

| Ref# | Title | Venue | Year | Author |
|---|---|---|---|---|
| [51] | "Variability Modeling for Cyber-Physical Production Systems: A Tertiary Literature Study" | JSS/IST | 2023 | 2nd |
| [140] | "Capabilities and Skills in Manufacturing: A Systematic Literature Study" | OJIES | 2023 | 1st |

Table 5.2: List of currently planned publications.

APPENDIX A

# Overview of Generative AI Tools Used

I used several human and software resources to correct and improve the manuscript. I used Grammarly extensively to correct the spelling and grammar. Furthermore, Grammarly contributed to this text by responding to these AI prompts (over the period of several months):

Prompts created by Grammarly

- “Improve it”

Prompts I wrote

- “Improve it keeping all Latex commands.”
- “Paraphrase it keeping all Latex commands.”
- “Shorten it”
- “Shorten it keeping the references.”
- “Shorten it keeping all Latex commands.”
- “Split this long sentence”
- “Shorten it more”
- “Summarize it keeping all Latex commands.”

# List of Figures

1.1 Illustrative car production example with its four dimensions of variability. 2
1.2 Rendered *shift forks* for a manual transmission with a list of parts. . . . . 6
1.3 Vertical and horizontal views of a *rocker switch* and its components [130]. 7
1.4 Traditional *basic planning* process of the initial CPPS design [87]. . . . . 9
1.5 Partial PPR model for production of a shift fork in VDI 3682 notation [199] with *products* as circles, production *processes* as rectangles, and production *resources* as rounded rectangles, from Meixner et al. [137]. . . . . . . . . . 10
1.6 Illustrative section of a potential shop floor layout with production process labels. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
1.7 Context of the contributions of the thesis in the Reference Architectural Model industrie 4.0 (RAMI 4.0), based on RAMI 4.0 [160]. . . . . . . . . . . . . 16
1.8 PPR model for a *welding process* in VDI 3682 [199] notation *without* (dashed connection *executes*) and *with* Capabilities and Skills (C&Ss), based on Meixner et al. [135]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
1.9 Feature model of a *shift fork* (cf. Section 1.1.1) [137] (cf. also Figure 3.13a). 19
1.10 Domain and Application Engineering (DAE) cycle in SPL and CPPS engineering [83, 152, 200]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
1.11 The applied Design Science approach in the context of this thesis [73, 74, 205]. 25

3.1 The publications forming this cumulative thesis' core (cf. Table 2.1) follow the three research lines: PKMO, ACEA, and IRVM approaches (cf. Chapter 1). Their relations are highlighted by specific arrows labeled with a research line. 44
3.2 Concept of a superimposed model for PPR models, in adapted VDI 3682 notation [199]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
3.3 The core entities *Product*, *Process*, and *Resource* and their inheritance hierarchy of the PPR knowledge model as Unified Modeling Language (UML) class diagram. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
3.4 Combined PPR knowledge model as UML class diagram. . . . . . . . . . 49
3.5 Domain model of the I4AN as UML class diagram. . . . . . . . . . . . . . 55
3.6 *Relation* entity and its inheritance hierarchy of the PPR and I4AN knowledge model as UML class diagram. . . . . . . . . . . . . . . . . . . . . . . . . 56
3.7 I4AN reference model (top row) and Patterns for Reuse (dashed contours) in an I4AN for the *shift fork* use case (cf. Section 1.1.1), in adapted VDI 3682 notation [199]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

3.8 The CSR framework for PPR models and artifacts with C&Ss, based on Domain and Application Engineering (DAE) [152] and the VDI 3695 [200, 83], from Meixner et al. [135]. . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
3.9 Rendered *shift forks* (in blue, magenta, and green) in a manual transmission, (Image: World Arm Lamp, CC0 on Wikipedia - `https://w.wiki/3DCf`), a section of the *shift fork* products in PPR–DSL, and the resulting feature model, from Meixner et al. [127]. . . . . . . . . . . . . . . . . . . . . . . . . 67
3.10 (Human & automated) EIPSE process steps [137] for exploring production process steps based on a product configuration (steps from IPSE [132] in dashed contours, updated steps with solid contours, and novel steps, additionally in a darker color), from Meixner et al. [137]. . . . . . . . . . . . . . . . . . . 69
3.11 Architecture of the EIPSE toolchain in UML component diagram notation (novel and updated components with solid contours), from Meixner et al. [137]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69
3.12 Conceptual mapping of the PPR–DSL knowledge model to Universal Variability Language (UVL) [192] for creating feature and decision models. . . 70
3.13 FeatureIDE feature models [116] of product (top) and production resource (bottom) variability of the *shift fork* use case. . . . . . . . . . . . . . . . . 71
3.14 After configuring the Product Feature Models (FM) in Step 3 of the EIPSE process (background) for the *shift fork* use case [130], the reduced Process Decision Models (DM) configuration is created in Step 4 of the EIPSE process using the EIPSE prototype's wizard (front), from Meixner et al. [137]. . . 72
3.15 During the configuration of the production steps in the Process DM in Step 5 of the EIPSE process, a suitable production process sequence for the *shift fork* use case [130] is defined. The EIPSE prototype provides a *rollback* option (right-hand side) for systematic process sequence exploration [133], from Meixner et al. [137]. . . . . . . . . . . . . . . . . . . . . . . . . . . 73

# List of Tables

1.1 Type Comparison Matrix (TCM) of *shift fork* types (columns) and their parts (rows). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
1.2 DOPLER DM [30] representing the *shift fork* product variability. . . . . . 20

2.1 Chronological list of core publications part of the thesis. . . . . . . . . . . 34
2.2 Research lines and artifacts mapped to CPPS *measures* and SPL *capabilities* with core thesis references. . . . . . . . . . . . . . . . . . . . . . . . . . . 35

3.1 Overview of the characteristics of the four case studies. . . . . . . . . . . 66
3.2 Excerpt of the generated DOPLER DM [30] representing the process variability of the *shift fork* case study. . . . . . . . . . . . . . . . . . . . . . . . . . . 72
3.3 "Time spent" by engineers with different backgrounds for EIPSE to the *chess piece* case study. E. . . Engineer, R. . . Researcher; ME. . . Mechanical engineering, SE. . . Systems engineering, from Meixner et al. [137]. . . . . . 74

5.1 Chronological list of additional publications not in focus of the cumulative thesis. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 205
5.2 List of currently planned publications. . . . . . . . . . . . . . . . . . . . . . 205

# Acronyms

**AAS** Asset Administration Shell. 54

**ACEA** Advanced CPPS Engineering Applications. 5, 24, 31, 33, 35, 41, 43, 44, 53, 79, 82–84, 209

**AML** AutomationML. 17

**BAPO** Business-Architecture-Process-Organization. 23

**C&S** Capability and Skill. 18, 28, 31, 35, 37, 38, 41, 42, 53, 56, 60–63, 79, 82, 209, 210

**CAD** Computer-Aided Design. 8, 9, 12, 55

**CDC** Cross-Discipline Constraint. 40, 70, 73

**CDL SQI** Christian Doppler Laboratory For Security and Quality Improvement in the Production System Lifecycle. 5, 26, 53

**CDL VaSiCS** Christian Doppler Laboratory For Mastering Variability in Software-intensive Cyber-Physical Production Systems. 78

**CPPS** Cyber-Physical Production System. ix, x, 1–15, 17–46, 48–50, 52–65, 67, 68, 70–86, 95, 96, 105, 115, 124, 134, 146, 157, 167, 178, 179, 209, 211, 213

**CPS** Cyber-Physical System. 22

**CSBE** Capability- and Skill-based Engineering. 17, 18, 27, 79, 82

**CSR** Capability and Skill Reuse. ix, 38, 42, 61, 63, 79, 80, 84, 210

**DAE** Domain and Application Engineering. 5, 19–21, 24, 28, 31, 38, 40, 41, 59–61, 63, 79, 209, 210

**DM** Decision Models. 20, 40, 41, 70–73, 75, 210, 211

**DSL** Domain-specific Language. 14, 18, 19, 28–30, 34–36, 41, 43, 48, 49, 52, 53, 62, 75, 79, 105, 157, 158

**EIPSE** Extended Iterative Process Sequence Exploration. x, 41, 64, 68–70, 72–76, 79–81, 84, 178, 179, 210, 211

**EMF** Eclipse Modeling Framework. 52

**ESPLA** Extractive Software Product Line Adoption. 6, 39, 68

**FM** Feature Models. 39–41, 69–73, 75, 76, 210

**FODA** Feature-Oriented Domain Analysis. 19

**FPD** Formalised Process Description. 17, 105

**I4AN** Industry 4.0 Asset Network. ix, 37, 38, 41, 53–60, 82, 84, 115, 124, 125, 209

**ICT** Information and Communication Technology. 1

**IPSE** Iterative Process Sequence Exploration. 40, 41, 64, 68, 69, 72, 73, 75, 79, 84, 167, 168, 210

**IRVM** Integrated Reuse and Variability Management. 5, 24, 31, 33, 35, 41, 43, 44, 68, 76, 79, 83, 84, 86, 95, 167, 178, 209

**LSP** Language Server Protocol. 53

**MBSE** Model-based Software Engineering. 5, 27

**MOF** Meta Object Facility. 18, 46

**OCL** Object Constraint Language. 22

**PDA** Project-Dependent Activity. 20, 61–63

**PDDL** Planning Domain Definition Language. 17

**PIA** Project-independent Activity. 20, 61, 63

**PKMO** Production Knowledge Models. 5, 23, 31, 33, 35, 41, 43, 44, 83, 84, 209

**PPR** Product-Process-Resource. ix, x, 1, 10, 13–18, 21, 23–25, 28–31, 33, 35–54, 56, 58, 61–66, 68, 71, 75, 78–86, 95, 96, 105, 106, 115, 124, 178, 209, 210

**PPR–DSL** Product-Process-Resource Domain-Specific Language. ix, 25, 31, 36, 39–41, 46, 48–53, 63, 65–71, 74–76, 78–81, 84, 105, 106, 157, 167, 178, 210

**PSE** Production Systems Engineering. 38, 115, 124, 125

**RAMI 4.0** Reference Architectural Model industrie 4.0. 15, 16, 209

**SDG** Sustainable Development Goal. 46, 78

**SLR** Systematic Literature Review. 19, 22

**SPL** Software Product Line. 3, 5, 13, 15, 19, 21–25, 27, 32, 33, 35–40, 43, 47, 64, 68, 76, 78, 80–82, 84, 86, 95, 96, 105, 115, 124, 134, 146, 157, 167, 178, 209, 211

**SQL** Structured Query Language. 52, 53

**TCM** Type Comparison Matrix. 8, 9, 17, 19, 45, 46, 63, 64, 67, 77, 211

**UML** Unified Modeling Language. 22, 46, 47, 49, 54–56, 69, 209, 210

**UVL** Universal Variability Language. 19, 68, 70, 71, 210

**VDI** Verein Deutscher Ingenieure (Association of German Engineers). 16, 17

**VERT** Variability Evolution Roundtrip Transformation. 39, 63–65, 146, 157

**VM** Variability Modeling. 86

**XML** Extensible Markup Language. 17, 44, 66

# Bibliography

[1] S. Ananieva, M. Kowal, T. Thüm, and I. Schaefer. Implicit constraints in partial feature models. In *7th Int. FOSD Workshop, FOSD@SPLASH 2016, Amsterdam, Netherlands, October 30, 2016*, pages 18–27, 2016.

[2] J. Andersson, R. de Lemos, S. Malek, and D. Weyns. *Modeling Dimensions of Self-Adaptive Software Systems*, pages 27–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-02161-9. doi: 10.1007/978-3-642-02161-9_2.

[3] S. Apel, D. Batory, C. Kaestner, and G. Saake. *Feature-Oriented Software Development: Concepts and Implementation*. Springer, 2013.

[4] P. Ardimento, N. Boffoli, and G. Superbo. Multi Software Product Lines: A Systematic Mapping Study. In *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE*, pages 470–476. INSTICC, SciTePress, 2020. ISBN 978-989-758-421-3. doi: 10.5220/0009469804700476.

[5] R. Bashroush, M. Garba, R. Rabiser, I. Groher, and G. Botterweck. CASE Tool Support for Variability Management in Software Product Lines. *ACM Computing Surveys*, 50(1):14:1–14:45, Mar. 2017.

[6] M. Becker, R. Rabiser, and G. Botterweck. Not Quite There Yet: Remaining Challenges in Systems and Software Product Line Engineering as Perceived by Industry Practitioners. In *SPLC (A)*, pages 179–190. ACM, 2024.

[7] L. Berardinelli, A. Mazak, O. Alt, M. Wimmer, and G. Kappel. Model-driven systems engineering: Principles and application in the CPPS domain. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*, pages 261–299. Springer, 2017.

[8] T. Berger, R. Rublack, D. Nair, J. M. Atlee, M. Becker, K. Czarnecki, and A. Wąsowski. A survey of variability modeling in industrial practice. In *7th International Workshop on Variability Modelling of Software-intensive Systems*, pages 7–14. ACM, 2013.

[9] T. Berger, J.-P. Steghöfer, T. Ziadi, J. Robin, and J. Martinez. The state of adoption and the challenges of systematic variability management in industry. *Empirical Software Engineering*, 25:1755–1797, 2020.

[10] S. Biffl, D. Gerhard, and A. Lüder. Introduction to the Multi-Disciplinary Engineering for Cyber-Physical Production Systems. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*, pages 1–24. Springer, 2017.

[11] S. Biffl, L. Kathrein, A. Lueder, **K. Meixner**, M. Sabou, L. Waltersdorfer, and D. Winkler. Software engineering risks from technical debt in the representation of product/ion knowledge. In A. Perkusich, editor, *The 31st International Conference on Software Engineering and Knowledge Engineering, SEKE 2019, Hotel Tivoli, Lisbon, Portugal, July 10-12, 2019*, volume 2019-July, pages 693–700. Knowledge Systems Institute Graduate School, 2019. doi: 10.18293/SEKE2019-037.

[12] S. Biffl, A. Lueder, **K. Meixner**, F. Rinker, M. Eckhart, and D. Winkler. Multi-view-model risk assessment in cyber-physical production systems engineering. In S. Hammoudi, L. F. Pires, E. Seidewitz, and R. Soley, editors, *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2021, Online Streaming, February 8-10, 2021*, pages 163–170. SCITEPRESS, 2021. doi: 10.5220/0010224801630170.

[13] S. Biffl, **K. Meixner**, D. Winkler, and A. Lueder. Towards efficient asset-based configuration management with a ppr asset directory. In *26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021*, volume 2021-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ETFA45728.2021.9613200.

[14] S. Biffl, J. Musil, A. Musil, **K. Meixner**, A. Lüder, F. Rinker, D. Weyns, and D. Winkler. An industry 4.0 asset-based coordination artifact for production systems engineering. In J. Almeida, D. Bork, G. Guizzardi, G. Guizzardi, M. Montali, H. Proper, and T. Sales, editors, *23rd IEEE Conference on Business Informatics, CBI 2021, Bolzano, Italy, September 1-3, 2021. Volume 1*, volume 2, pages 92–101. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/CBI52690.2021.00020.

[15] S. Biffl, S. Kropatschek, E. Kiesling, **K. Meixner**, and A. Lueder. Risk-driven derivation of operation checklists from multi-disciplinary engineering knowledge. In *20th IEEE International Conference on Industrial Informatics, INDIN 2022, Perth, Australia, July 25-28, 2022*, volume 2022-July, pages 7–14. Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/INDIN51773.2022.9976096.

[16] S. Biffl, **K. Meixner**, D. Hoffmann, J. Musil, H. Rahmani, and A. Lueder. Towards coordinating production reconfiguration. In *27th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2022, Stuttgart, Germany,*

*September 6-9, 2022*, volume 2022-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ETFA52439.2022.9921665.

[17] S. Biffl, D. Hoffmann, E. Kiesling, **K. Meixner**, A. Lueder, and D. Winkler. Validating production test scenarios with cyber-physical system design models. In *25th IEEE Conference on Business Informatics, CBI 2023 - Volume 1, Prague, Czech Republic, June 21-23, 2023*, pages 1–10. Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/CBI58679.2023.10187499.

[18] S. Biffl, **K. Meixner**, D. Hoffmann, D. Winkler, and A. Lueder. Towards test-driven performance validation of a flexible cyber-physical production system. In *28th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2023, Sinaia, Romania, September 12-15, 2023*, volume 2023-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ETFA54631.2023.10275573.

[19] J. Bosch. Software Product Line Engineering. In R. Capilla, J. Bosch, and K.-C. Kang, editors, *Systems and Software Variability Management SE - 1*, pages 3–24. Springer Berlin Heidelberg, 2013.

[20] C. Brink, M. Peters, and S. Sachweh. Configuration of mechatronic multi product lines. In *Proc. of the 3rd Int'l Workshop on Variability & Composition*, VariComp '12, pages 7–11, New York, NY, USA, 2012. ACM.

[21] C. Burger. Model Difference Analysis for CPPS Engineering Models with Variability. Master's thesis, TU Wien, Wien, 2022.

[22] B. Caesar, M. Nieke, A. Köcher, C. Hildebrandt, C. Seidl, A. Fay, and I. Schaefer. Context-sensitive reconfiguration of collaborative manufacturing systems. *IFAC-PapersOnLine*, 52(13):307 – 312, 2019. ISSN 2405-8963. 9th IFAC Conf. on Manufacturing Modelling, Management and Control.

[23] G. H. Campbell, S. R. Faulk, and D. M. Weiss. Introduction To Synthesis. Technical report, INTRO_SYNTHESIS_PROCESS-90019-N, Software Productivity Consortium, Herndon, VA, USA, 1990.

[24] L. Chen and M. A. Babar. A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology*, 53(4):344–362, Apr. 2011.

[25] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. SEI series in software engineering. Addison-Wesley, 2002. ISBN 978-0-201-70332-0.

[26] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged configuration using feature models. In *3rd International Conference on Software Product Lines*, pages 266–283, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-28630-1. doi: https://doi.org/10.1007/978-3-540-28630-1_17.

[27] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged configuration through specialization and multilevel configuration of feature models. *Software process: improvement and practice*, 10(2):143–169, 2005.

[28] K. Czarnecki, S. Helsen, and U. Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software process: Improvement and practice*, 10 (1):7–29, 2005.

[29] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wąsowski. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. In *6th International Workshop on Variability Modeling of Software-Intensive Systems*, pages 173–182. ACM, 2012.

[30] D. Dhungana, P. Grünbacher, and R. Rabiser. The DOPLER Meta-Tool for Decision-Oriented Variability Modeling: A Multiple Case Study. *Automated Software Engineering*, 18(1):77–114, 2011.

[31] DIN8580. DIN 8580 – Manufacturing processes. German Institute for Standardisation, 2022.

[32] R. Drath. *Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA*. Springer-Verlag, 2009.

[33] R. Drath, A. Luder, J. Peschke, and L. Hundt. Automationml - the glue for seamless automation engineering. In *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, pages 616–623, 2008.

[34] M. Eckhart, **K. Meixner**, D. Winkler, and A. Ekelhart. Securing the testing process for industrial automation software. *Computers and Security*, 85:156–180, 2019. doi: 10.1016/j.cose.2019.04.016.

[35] C. Engelbrecht. Coordinated Multi-View Graph Analysis and Improvement with Applications in Cyber-Physical Production Systems Engineering. Master's thesis, TU Wien, Wien, 2021.

[36] E. Engström, M.-A. Storey, P. Runeson, M. Höst, and M. T. Baldassarre. How software engineering research aligns with design science: a review. *Empirical Software Engineering*, 25:2630–2660, 2020.

[37] J. Epp, T. Robert, O. Ruch, and A. Olechowski. Towards SysML v2 as a Variability Modeling Language. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 251–256, 2023. doi: 10.1109/MODELS-C59198.2023.00054.

[38] H. S. Fadhlillah, K. Feichtinger, **K. Meixner**, L. Sonnleithner, R. Rabiser, and A. Zoitl. Towards multidisciplinary delta-oriented variability management in cyber-physical production systems. In P. Arcaini, X. Devroey, and A. Fantechi, editors,

*VaMoS '22: 16th International Working Conference on Variability Modelling of Software-Intensive Systems, Florence, Italy, February 23 - 25, 2022*, pages 13:1–13:10. Association for Computing Machinery, 2022. doi: 10.1145/3510466.3511273.

[39] H. S. Fadhlillah, K. Feichtinger, **K. Meixner**, L. Sonnleithner, R. Rabiser, and A. Zoitl. Towards multidisciplinary delta-oriented variability management in cyber-physical production systems. In *Proc. of the 16th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS)*. ACM, 2022. ISBN 9781450396042.

[40] M. Fang. *Model-Based Software Derivation for Industrial Automation Management Systems*. PhD thesis, Technische Universität Kaiserslautern, 2019.

[41] K. Feichtinger. *A Flexible Approach For Transforming Variability Artifacts*. phdthesis, Johannes Kepler University, 2023. URL `https://epub.jku.at/obvulihs/content/titleinfo/9132023`.

[42] K. Feichtinger and R. Rabiser. Variability model transformations: Towards unifying variability modeling. In *46th Euromicro Conference on Software Engineering and Advanced Applications*, Portoroz, Slovenia, 2020. IEEE.

[43] K. Feichtinger, **K. Meixner**, R. Rabiser, and S. Biffl. Variability transformation from industrial engineering artifacts: An example in the cyber-physical production systems domain. In R. Capilla, P. Collet, P. Gazzillo, J. Krueger, R. E. Lopez-Herrejon, S. Nadi, G. Perrouin, I. Reinhartz-Berger, J. Rubin, and I. Schaefer, editors, *SPLC '20: 24th ACM International Systems and Software Product Line Conference, Montreal, Quebec, Canada, October 19-23, 2020, Volume B*, volume Part F164402-B, pages 65–73. Association for Computing Machinery, 2020. doi: 10.1145/3382026.3425770.

[44] K. Feichtinger, **K. Meixner**, R. Rabiser, and S. Biffl. Variability Transformation from Industrial Engineering Artifacts: An Example in the Cyber-Physical Production Systems Domain. In *3rd International Workshop on Variability and Evolution of Software-Intensive Systems (VariVolution), SPLC '20: 24th ACM International Systems and Software Product Line Conference, Volume B*, pages 65–73. ACM, 2020.

[45] K. Feichtinger, **K. Meixner**, R. Rabiser, and S. Biffl. A systematic study as foundation for a variability modeling body of knowledge. In S. A. Baldassarre M.T., Scanniello G., editor, *47th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2021, Palermo, Italy, September 1-3, 2021*, pages 25–28. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/SEAA53835.2021.00012.

[46] K. Feichtinger, J. Stöbich, D. Romano, and R. Rabiser. Travart: An approach for transforming variability models. In *15th International Working Conference on*

*Variability Modelling of Software-Intensive Systems*, pages 8:1–8:10. ACM, 2021. ISBN 9781450388245.

[47] K. Feichtinger, **K. Meixner**, S. Biffl, and R. Rabiser. Evolution support for custom variability artifacts using feature models: A study in the cyber-physical production systems domain. In G. Perrouin, N. Moha, and A.-D. Seriai, editors, *Reuse and Software Quality - 20th International Conference on Software and Systems Reuse, ICSR 2022, Montpellier, France, June 15-17, 2022, Proceedings*, volume 13297 LNCS of *Lecture Notes in Computer Science*, pages 79–84. Springer Science and Business Media Deutschland GmbH, 2022. doi: 10.1007/978-3-031-08129-3_5.

[48] K. Feichtinger, **K. Meixner**, F. Rinker, I. Koren, H. Eichelberger, T. Heinemann, J. Holtmann, M. Konersmann, J. Michael, E.-M. Neumann, J. Pfeiffer, R. Rabiser, M. Riebisch, and K. Schmid. Industry voices on software engineering challenges in cyber-physical production systems engineering. In *27th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2022, Stuttgart, Germany, September 6-9, 2022*, volume 2022-September, pages 1–8. Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ETFA52439.2022.9921568.

[49] K. Feichtinger, **K. Meixner**, F. Rinker, I. Koren, H. Eichelberger, T. Heinemann, J. Holtmann, M. Konersmann, J. Michael, E.-M. Neumann, J. Pfeiffer, R. Rabiser, M. Riebisch, and K. Schmid. Industry voices on software engineering challenges in cyber-physical production systems engineering. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2022.

[50] K. Feichtinger, **K. Meixner**, F. P. Rinker, I. Koren, H. Eichelberger, T. Heinemann, J. Holtmann, M. Konersmann, J. Michael, E.-M. Neumann, J. Pfeiffer, R. Rabiser, M. Riebisch, and K. Schmid. Software in Cyberphysischen Produktionssystemen: Herausforderungen zur Umsetzung in der Industrie. *atp magazin*, 65(4):62–68, 2023. doi: 10.17560/atp.v65i4.2646.

[51] K. Feichtinger, **K. Meixner**, H. S. Fadhlillah, and R. Rabiser. Variability Modeling for Cyber-Physical Production Systems: A Tertiary Literature Study (planned). *JSS/IST*, pages 1–8, 2025.

[52] M. Felderer and G. H. Travassos, editors. *Contemporary Empirical Methods in Software Engineering*. Springer, 2020.

[53] Y. Fidan, A. Lüder, **K. Meixner**, L. Baumann, and J. Arlinghaus. Decision support for frugal products and production systems based on product-process-resource-skill & variability models. In M. D., editor, *Procedia CIRP*, volume 104, pages 1619–1625. Elsevier B.V., 2021. doi: 10.1016/j.procir.2021.11.273.

[54] S. Fischer, L. Linsbauer, R. E. Lopez-Herrejon, and A. Egyed. Enhancing Clone-and-Own with Systematic Reuse for Developing Software Variants. In *2014 IEEE*

*International Conference on Software Maintenance and Evolution*, pages 391–400, sep 2014. doi: 10.1109/ICSME.2014.61.

[55] M. Forlingieri. The four dimensions of variability and their impact on MBPLE: how to approach variability in the development of aircraft product lines at airbus. In *16th International Working Conference on Variability Modelling of Software-Intensive Systems*, pages 15:1–15:4, New York, NY, USA, 2022. ACM. doi: https://doi.org/10.1145/3510466.3511275.

[56] M. Fowler. *Domain-Specific Languages*. Pearson Education, 2010.

[57] M. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2012.

[58] R. Froschauer, A. Koecher, **K. Meixner**, S. Schmitt, and F. Spitzer. Capabilities and skills in manufacturing: A survey over the last decade of etfa. In *27th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2022, Stuttgart, Germany, September 6-9, 2022*, volume 2022-September, pages 1–8. Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ETFA52439.2022.9921560.

[59] M. Galster, D. Weyns, D. Tofan, B. Michalik, and P. Avgeriou. Variability in Software Systems - A Systematic Literature Review. *IEEE Transactions on Software Engineering*, 40(3):282–306, Mar. 2014. ISSN 0098-5589.

[60] L. Gerling, S. Greiner, **K. Meixner**, and G. K. Michelon. Fourth international workshop on variability and evolution of software-intensive systems (varivolution 2021). In M. R. Mousavi and P.-Y. Schobbens, editors, *SPLC '21: 25th ACM International Systems and Software Product Line Conference, Leicester, United Kingdom, September 6-11, 2021, Volume A*, volume Part F171624-A, page 204. Association for Computing Machinery, 2021. doi: 10.1145/3461001.3473055.

[61] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language. Technical report, Technical report CVC TR-98-003/DCS TR-1165, Yale, 1998.

[62] S. Greiner, **K. Meixner**, G. K. Michelon, and P. Collet. Fifth international workshop on variability and evolution of software-intensive systems (varivolution 2022). In A. Felfernig, L. Fuentes, J. Cleland-Huang, W. K. G. Assunccao, A. A. Falkner, M. Azanza, M. a. Rodriguez Luaces, M. Bhushan, L. Semini, X. Devroey, C. M. L. Werner, C. Seidl, V.-M. Le, and J. M. Horcas, editors, *SPLC '22: 26th ACM International Systems and Software Product Line Conference, Graz, Austria, September 12 - 16, 2022, Volume A*, volume A, page 263. Association for Computing Machinery, Inc, 2022. doi: 10.1145/3546932.3547018.

[63] S. Greiner, X. Ternava, **K. Meixner**, and S. Krieter. Sixth international workshop on variability and evolution of software-intensive systems (varivolution 2023). In P. Arcaini, M. H. ter Beek, G. Perrouin, I. Reinhartz-Berger, M. R. Luaces,

C. Schwanninger, S. Ali, M. Varshosaz, A. Gargantini, S. Gnesi, M. Lochau, L. Semini, and H. Washizaki, editors, *Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume A, SPLC 2023, Tokyo, Japan. 28 August 2023- 1 September 2023*, volume A-1, page 274. Association for Computing Machinery, 2023. doi: 10.1145/3579027.3609003.

[64] S. Greiner, B. Wiesmayr, K. Feichtinger, **K. Meixner**, M. Konersmann, J. Pfeiffer, M. Oberlehner, D. Schmalzing, A. Wortmann, B. Rumpe, R. Rabiser, and A. Zoitl. Maturity evaluation of domain-specific language ecosystems for cyber-physical production systems. In *28th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2023, Sinaia, Romania, September 12-15, 2023*, volume 2023-September, pages 1–8. Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ETFA54631.2023.10275624.

[65] S. Greiner, B. Wiesmayr, K. Feichtinger, **K. Meixner**, M. Konersmann, J. Pfeiffer, M. Oberlehner, D. Schmalzing, A. Wortmann, B. Rumpe, R. Rabiser, and A. Zoitl. Dataset: Maturity evaluation of domain-specific language ecosystems for cyber-physical production systems (version 1). `https://doi.org/10.5281/zenodo.7882951`, July 2023. Accessed on YYYY-MM-DD.

[66] S. Greiner, K. Schmid, T. Berger, S. Krieter, and **K. Meixner**. Generative AI and software variability - A research vision. In T. Kehrer, M. Huchard, L. Teixeira, and C. Birchler, editors, *Proceedings of the 18th International Working Conference on Variability Modelling of Software-Intensive Systems, VaMoS 2024, Bern, Switzerland, February 7-9, 2024*, pages 71–76. ACM, 2024. doi: 10.1145/3634713.3634722.

[67] S. Grüner, A. Burger, T. Kantonen, and J. Rückert. Incremental migration to software product line engineering. In *Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A - Volume A*, SPLC '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375696. doi: 10.1145/3382025.3414956.

[68] P. Grünbacher, R. Rabiser, D. Dhungana, and M. Lehofer. Model-based Customization and Deployment of Eclipse-Based Tools: Industrial Experiences. In *24th IEEE/ACM International Conference on Automated Software Engineering*, pages 247–256. IEEE/ACM, 2009.

[69] V. Gunes, S. Peter, T. Givargis, and F. Vahid. A survey on concepts, applications, and challenges in cyber-physical systems. *KSII Transactions on Internet & Information Systems*, 8(12), 2014.

[70] K. W. Helbing. *Typenvertreter*, pages 1423–1428. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018. ISBN 978-3-662-55551-4. doi: 10.1007/978-3-662-55551-4_66.

[71] J. Herzog. *Entwicklung einer fähigkeitsbasierten Planungsmethode zur Analyse der Wiederverwendbarkeit von Anlagen am Beispiel von Schraubprozessen in der*

*Automobilendmontage (Development of a capability-based planning method for analyzing the reusability of systems using the example of screwdriving processes in final automotive assembly)*. phdthesis, Otto-von-Guericke University, 2023.

[72] J. Herzog, H. Röpke, and A. Lüder. Allocation of PPRS for the plant planning in the final automotive assembly. In *2020 25th IEEE ETFA*, pages 813–820. IEEE, 2020.

[73] A. R. Hevner. A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4, 2007.

[74] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *Management Information Systems Quarterly*, 28(1):6, 2004.

[75] B. Hjørland. Domain analysis in information science: Eleven approaches–traditional as well as innovative. *Journal of Documentation*, 58(4):422–462, 2002. doi: 10.1108/00220410210431136.

[76] D. Hoffmann, S. Biffl, **K. Meixner**, and A. Lueder. Towards design patterns for production security. In *27th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2022, Stuttgart, Germany, September 6-9, 2022*, volume 2022-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ETFA52439.2022.9921691.

[77] D. Hoffmann, N. Nowacki, S. Biffl, E. Kiesling, **K. Meixner**, and A. Lüder. Interdisciplinary production risk exploration: A grounded approach to integrate data- and knowledge-driven analytics. *Procedia CIRP*, 120:1016–1021, 2023. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2023.09.117. 56th CIRP International Conference on Manufacturing Systems 2023.

[78] G. Holl, P. Grünbacher, and R. Rabiser. A Systematic Review and an Expert Survey on Capabilities Supporting Multi Product Lines. *Information and Software Technology*, 54(8):828–852, 2012.

[79] A. Hubaux, A. Classen, and P. Heymans. Formal modelling of feature configuration workflows. In *SPLC*, volume 446 of *ACM International Conference Proceeding Series*, pages 221–230. ACM, 2009.

[80] A. Hubaux, M. Acher, T. T. Tun, P. Heymans, P. Collet, and P. Lahire. *Separating Concerns in Feature Models: Retrospective and Support for Multi-Views*, pages 3–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-36654-3".

[81] D. G. D. L. Iglesia and D. Weyns. Mape-k formal templates to rigorously design behaviors for self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems*, 10(3), Sept. 2015. ISSN 1556-4665.

[82] International Electrotechnical Commission (IEC), TC65/WG6. IEC 61499-1, Function Blocks - part 1: Architecture: Edition 2.0, 2012.

[83] N. Jazdi, C. Maga, P. Göhner, T. Ehben, T. Tetzner, and U. Löwen. Mehr Systematik für den Anlagenbau und das industrielle Lösungsgeschäft - Gesteigerte Effizienz durch Domain Engineering. *Automatisierungstechnik*, 58(9):524–532, 2010. doi: doi:10.1524/auto.2010.0867.

[84] E. Järvenpää, N. Siltala, O. Hylli, and M. Lanz. The development of an ontology for describing the capabilities of manufacturing resources. *Journal of Intelligent Manufacturing*, 30:959–978, Feb. 2019. ISSN 0956-5515. doi: 10.1007/s10845-018-1427-6.

[85] E. Järvenpää, N. Siltala, O. Hylli, and M. Lanz. Implementation of capability matchmaking software facilitating faster production system design and reconfiguration planning. *Journal of Manufacturing Systems*, 53:261–270, 2019. ISSN 1878-6642. doi: 10.1016/j.jmsy.2019.10.003.

[86] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical report, Carnegie-Mellon Univ., Pittsburgh, Pa, Software Engineering Inst., 1990.

[87] L. Kathrein, A. Lueder, **K. Meixner**, D. Winkler, and S. Biffl. *Product/ion-Aware Analysis of Collaborative Systems Engineering Processes*, pages 151–185. Springer International Publishing, 2019. doi: 10.1007/978-3-030-25312-7_7.

[88] L. Kathrein, A. Lueder, **K. Meixner**, D. Winkler, and S. Biffl. Production-aware analysis of multi-disciplinary systems engineering processes. In J. Filipe, M. Smialek, A. Brodsky, and S. Hammoudi, editors, *Proceedings of the 21st International Conference on Enterprise Information Systems, ICEIS 2019, Heraklion, Crete, Greece, May 3-5, 2019, Volume 2*, volume 2, pages 48–60. SciTePress, 2019. doi: 10.5220/0007618000480060.

[89] L. Kathrein, **K. Meixner**, D. Winkler, A. Lueder, and S. Biffl. Efficient production system resource exploration considering product/ion requirements. In *24th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2019, Zaragoza, Spain, September 10-13, 2019*, volume 2019-September, pages 665–672. Institute of Electrical and Electronics Engineers Inc., 2019. doi: 10.1109/ETFA.2019.8869499.

[90] L. Kathrein, **K. Meixner**, D. Winkler, A. Lueder, and S. Biffl. A meta-model for representing consistency as extension to the formal process description. In *24th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2019, Zaragoza, Spain, September 10-13, 2019*, volume 2019-September, pages 1653–1656. Institute of Electrical and Electronics Engineers Inc., 2019. doi: 10.1109/ETFA.2019.8869071.

[91] L. Kathrein, **K. Meixner**, D. Winkler, A. Lueder, and S. Biffl. Extending the formal process description towards consistency in product/ion-aware modeling. In *24th IEEE International Conference on Emerging Technologies and Factory*

*Automation, ETFA 2019, Zaragoza, Spain, September 10-13, 2019*, volume 2019-September, pages 679–686. Institute of Electrical and Electronics Engineers Inc., 2019. doi: 10.1109/ETFA.2019.8869006.

[92] L. Kathrein, **K. Meixner**, D. Winkler, A. Lueder, and S. Biffl. Product/ion-aware modeling approaches that support tracing design decisions. In *17th IEEE International Conference on Industrial Informatics, INDIN 2019, Helsinki, Finland, July 22-25, 2019*, volume 2019-July, pages 118–125. Institute of Electrical and Electronics Engineers Inc., 2019. doi: 10.1109/INDIN41052.2019.8972147.

[93] N. Keddis, G. Kainz, and A. Zoitl. Capability-based planning and scheduling for adaptable manufacturing systems. In *2014 19th IEEE ETFA*, pages 1–8, Sep. 2014. doi: 10.1109/ETFA.2014.7005213.

[94] B. Kiagho, R. Machunda, A. Hilonga, and K. Njau. Performance of water filters towards the removal of selected pollutants in arusha, tanzania. *Tanzania Journal of Science*, 42(1):134–147, 2016.

[95] A. Köcher, C. Hildebrandt, B. Caesar, J. Bakakeu, J. Peschke, A. Scholz, and A. Fay. Automating the Development of Machine Skills and their Semantic Description. In *25th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA 2020, Vienna, Austria, September 8-11, 2020*, pages 1013–1018. IEEE, 2020. doi: 10.1109/etfa46521.2020.9211933.

[96] A. Köcher, C. Hildebrandt, L. M. V. da Silva, and A. Fay. A Formal Capability and Skill Model for Use in Plug and Produce Scenarios. In *25th IEEE Int. Conf. on Emerging Technologies and Factory Automation, ETFA 2020, Vienna, Austria, September 8-11, 2020*, pages 1663–1670. IEEE, 2020. doi: 10.1109/etfa46521.2020.9211874.

[97] A. Koecher, A. Belyaev, J. Hermann, J. Bock, **K. Meixner**, M. Volkmann, M. Winter, P. Zimmermann, S. Grimm, and C. Diedrich. A reference model for common understanding of capabilities and skills in manufacturing [ein referenzmodell für ein gemeinsames verständnis von capabilities und skills von anlagen]. *At-Automatisierungstechnik*, 71(2):94–104, 2023. doi: 10.1515/auto-2022-0117.

[98] I. Koren, F. Rinker, **K. Meixner**, M. Kroeger, and M. Zeng. Implementing devops practices in cpps using microservices and gitops. In *28th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2023, Sinaia, Romania, September 12-15, 2023*, volume 2023-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ETFA54631.2023.10275433.

[99] I. Koren, F. Rinker, **K. Meixner**, J. Matevska, and J. Walter. Challenges and opportunities of devops in cyber-physical production systems engineering. In *6th IEEE International Conference on Industrial Cyber-Physical Systems, ICPS 2023,*

*Wuhan, China, May 8-11, 2023*, pages 1–6. Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICPS58381.2023.10128073.

[100] D. Kretz. Improving Model Reviewing and Experimentation with Tool Support: A controlled experiment. Master's thesis, TU Wien, Wien, 2021.

[101] S. Kropatschek, T. Steuer, E. Kiesling, **K. Meixner**, T. Fruehwirth, P. Sommer, D. Schachinger, and S. Biffl. Towards the representation of cross-domain quality knowledge for efficient data analytics. In *26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021*, volume 2021-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ETFA45728.2021.9613406.

[102] S. Kropatschek, O. Gert, I. Ayatollahi, **K. Meixner**, E. Kiesling, A. Steigberger, A. Lueder, and S. Biffl. Designing a Digital Shadow for Efficient, Low-Delay Analysis of Production Quality Risk. In *27th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2022, Stuttgart, Germany, September 6-9, 2022*, volume 2022-September, pages 1–8. Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ETFA52439.2022.9921582.

[103] S. Kropatschek, T. Steuer, E. Kiesling, **K. Meixner**, I. Ayatollahi, P. Sommer, and S. Biffl. Analysis of Quality Issues in Production With Multi-view Coordination Assets. In A. Bernard, A. Dolgui, H. Benderbal, D. Ivanov, D. Lemoine, and F. Sgarbossa, editors, *10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022: Nantes, France, 22-24 June 2022*, volume 55, pages 2938–2943. Elsevier B.V., 2022. doi: 10.1016/j.ifacol.2022.10.178.

[104] J. Krüger, S. Nielebock, S. Krieter, C. Diedrich, T. Leich, G. Saake, S. Zug, and F. Ortmeier. Beyond Software Product Lines: Variability Modeling in Cyber-Physical Systems. In *21st International Systems and Software Product Line Conference, SPLC 2017, Volume A, Sevilla, Spain, September 25-29, 2017*, pages 237–241, New York, NY, USA, 2017. ACM.

[105] S. Lee. Disassembly planning based on subassembly extraction. In *Third ORSA/-TIMS Conference on Flexible Manufacturing System*, pages 383–388, 1989.

[106] L. Linsbauer, P. Westphal, P. M. Bittner, S. Krieter, T. Thüm, and I. Schaefer. Derivation of subset product lines in featureide. In *26th ACM International Systems and Software Product Line Conference - Volume B*, SPLC '22, page 38–41, New York, NY, USA, 2022. ACM. doi: 10.1145/3503229.3547033.

[107] S. Lity, S. Nahrendorf, T. Thüm, C. Seidl, and I. Schaefer. 175% Modeling for Product-Line Evolution of Domain Artifacts. In R. Capilla, M. Lochau, and L. Fuentes, editors, *the 12th VAMOS 2018, Madrid, Spain, February 7-9, 2018*, pages 27–34. ACM, 2018. ISBN 978-1-4503-5398-4.

[108] M. Lochau, J. Bürdek, S. Hölzle, and A. Schürr. Specification and automated validation of staged reconfiguration processes for dynamic software product lines. *Software and Systems Modeling*, 16(1):125–152, 2017.

[109] A. Lüder, N. Schmidt, K. Hell, H. Röpke, and J. Zawisza. *Fundamentals of Artifact Reuse in CPPS*, pages 113–138. Springer International Publishing, Cham, 2017. ISBN 978-3-319-56345-9. doi: 10.1007/978-3-319-56345-9_5.

[110] A. Lüder, S. Biffl, D. Hoffmann, and **K. Meixner**. Identifying required Knowledge for Production System Digitalization projects. In *29th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2024, Padova, Italy, September 10-13, 2024 (in press)*, pages 1–8. IEEE, 2024. doi: 10.1109/ETFA61755.2024.10710645.

[111] A. Lüder, D. Hoffmann, R. Gudder, S. Biffl, and **K. Meixner**. Representing Property Dependencies within AutomationML Based Digital Twins. In *29th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2024, Padova, Italy, September 10-13, 2024*, pages 1–4. IEEE, 2024. doi: 10.1109/ETFA61755.2024.10710983.

[112] A. Lueder, **K. Meixner**, A.-K. Behnert, and S. Biffl. Modelling engineered object dependencies in an automationml-based tool chain. In *26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021*, volume 2021-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ETFA45728.2021.9613530.

[113] A. Lueder, D. Hoffmann, **K. Meixner**, P. Huenecke, and S. Biffl. Consistent extension of networks of digital representations of production system assets. In *28th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2023, Sinaia, Romania, September 12-15, 2023*, volume 2023-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ETFA54631.2023.10275688.

[114] A. Lüder, **K. Meixner**, and S. Biffl. Engineering data treasures, their collection and use. In A. Bernard, A. Dolgui, H. Benderbal, D. Ivanov, D. Lemoine, and F. Sgarbossa, editors, *IFAC-PapersOnLine*, volume 55, pages 2623–2628. Elsevier B.V., 2022. doi: 10.1016/j.ifacol.2022.10.105.

[115] J. Martinez, W. K. Assunção, and T. Ziadi. ESPLA: A catalog of Extractive SPL Adoption case studies. In *21st International Systems and Software Product Line Conference*, pages 38–41. ACM, 2017.

[116] J. Meinicke, T. Thüm, R. Schröter, F. Benduhn, T. Leich, and G. Saake. *Mastering Software Variability with FeatureIDE*. Springer, 2017.

[117] **K. Meixner**. Integrating variability modeling of products, processes, and resources in cyber-physical production systems engineering. In R. Capilla, P. Collet, P. Gazzillo, J. Krueger, R. E. Lopez-Herrejon, S. Nadi, G. Perrouin, I. Reinhartz-Berger, J. Rubin, and I. Schaefer, editors, *SPLC '20: 24th ACM International Systems and Software Product Line Conference, Montreal, Quebec, Canada, October 19-23, 2020, Volume B*, volume Part F164402-B, pages 96–103. Association for Computing Machinery, 2020. doi: 10.1145/3382026.3431247.

[118] **K. Meixner**, R. Rabiser, and S. Biffl. Towards modeling variability of products, processes and resources in cyber-physical production systems engineering. In C. Cetina, O. Diaz, L. Duchien, M. Huchard, R. Rabiser, C. Salinesi, C. Seidl, X. Tërnava, L. Teixeira, T. Thuem, and T. Ziadi, editors, *Proceedings of the 23rd International Systems and Software Product Line Conference, SPLC 2019, Volume B, Paris, France, September 9-13, 2019*, volume B, pages 68:1–68:8. Association for Computing Machinery, 2019. doi: 10.1145/3307630.3342411.

[119] **K. Meixner**, D. Winkler, and S. Biffl. Towards combined process & tool variability management in software testing. In D. Weyns and G. Perrouin, editors, *Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS 2019, Leuven, Belgium, February 06-08, 2019*, pages 5:1–5:6. Association for Computing Machinery, 2019. doi: 10.1145/3302333.3302339.

[120] **K. Meixner**, D. Winkler, and S. Biffl. Supporting domain experts by using model-based equivalence class partitioning for efficient test data generation. In *24th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2019, Zaragoza, Spain, September 10-13, 2019*, volume 2019-September, pages 134–141. Institute of Electrical and Electronics Engineers Inc., 2019. doi: 10.1109/ETFA.2019.8869145.

[121] **K. Meixner**, D. Winkler, P. Novak, and S. Biffl. Towards model-driven verification of robot control code using abstract syntax trees in production systems engineering. In S. B., editor, *Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2019, Prague, Czech Republic, February 20-22, 2019*, pages 402–409. Science and Technology Publications, Lda, 2019. doi: 10.5220/0007484104020409.

[122] **K. Meixner**, D. Winkler, M. Wapp, R. Rosendahl, and S. Biffl. Investigating the performance of selected data storage concepts for automationml models. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society, Lisbon, Portugal, October 14-17, 2019*, volume 2019-October, pages 2785–2791. IEEE Computer Society, 2019. doi: 10.1109/IECON.2019.8927275.

[123] **K. Meixner**, J. Decker, H. Marcher, A. Lueder, and S. Biffl. Towards a domain-specific language for product-process-resource constraints. In *25th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2020, Vienna, Austria, September 8-11, 2020*, volume 2020-September, pages

1405–1408. Institute of Electrical and Electronics Engineers Inc., 2020. doi: 10.1109/ETFA46521.2020.9212063.

[124] **K. Meixner**, L. Kathrein, D. Winkler, A. Lueder, and S. Biffl. Efficient test case generation from product and process model properties and preconditions. In *25th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2020, Vienna, Austria, September 8-11, 2020*, volume 2020-September, pages 859–866. Institute of Electrical and Electronics Engineers Inc., 2020. doi: 10.1109/ETFA46521.2020.9212003.

[125] **K. Meixner**, A. Lueder, J. Herzog, H. Roepke, and S. Biffl. Modeling expert knowledge for optimal cpps resource selection for a product portfolio. In *25th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2020, Vienna, Austria, September 8-11, 2020*, volume 2020-September, pages 1687–1694. Institute of Electrical and Electronics Engineers Inc., 2020. doi: 10.1109/ETFA46521.2020.9212067.

[126] **K. Meixner**, R. Rabiser, and S. Biffl. Feature identification for engineering model variants in cyber-physical production systems engineering. In M. Cordy, M. Acher, D. Beuche, and G. Saake, editors, *VaMoS '20: 14th International Working Conference on Variability Modelling of Software-Intensive Systems, Magdeburg Germany, February 5-7, 2020*, pages 18:1–18:5. Association for Computing Machinery, 2020. doi: 10.1145/3377024.3377043.

[127] **K. Meixner**, K. Feichtinger, R. Rabiser, and S. Biffl. A reusable set of real-world product line case studies for comparing variability models in research and practice. In M. Mousavi, P.-Y. Schobbens, H. Araujo, I. Schaefer, M. ter Beek, X. Devroey, J. Rojas, R. Rabiser, M. Varshosaz, T. Kishi, and J. Lee, editors, *SPLC '21: 25th ACM International Systems and Software Product Line Conference, Leicester, United Kindom, September 6-11, 2021, Volume B*, volume Part F171625-B, pages 105–112. Association for Computing Machinery, 2021. doi: 10.1145/3461002.3473946.

[128] **K. Meixner**, A. Lueder, J. Herzog, D. Winkler, and S. Biffl. Patterns for reuse in production systems engineering. In S.-K. Chang, editor, *The 33rd International Conference on Software Engineering and Knowledge Engineering, SEKE 2021, KSIR Virtual Conference Center, USA, July 1 - July 10, 2021*, pages 1623–1659. KSI Research Inc., 2021. doi: 10.18293/SEKE2021-150.

[129] **K. Meixner**, A. Lüder, J. Herzog, D. Winkler, and S. Biffl. Patterns for reuse in production systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 31(11-12):1623–1659, 2021. doi: 10.1142/S0218194021400155.

[130] **K. Meixner**, F. Rinker, H. Marcher, J. Decker, and S. Biffl. A domain-specific language for product-process-resource modeling. In *26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras,*

*Sweden, September 7-10, 2021*, volume 2021-September, pages 1–8. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ETFA45728.2021.9613674.

[131] **K. Meixner**, F. Rinker, H. Marcher, J. Decker, and S. Biffl. A Domain-Specific Language for Product-Process-Resource Modeling. In *26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021*, pages 1–8. IEEE, 2021.

[132] **K. Meixner**, K. Feichtinger, R. Rabiser, and S. Biffl. Efficient production process variability exploration. In F. A. Arcaini P., Devroey X., editor, *VaMoS '22: 16th International Working Conference on Variability Modelling of Software-Intensive Systems, Florence, Italy, February 23 - 25, 2022*, pages 14:1–14:9. Association for Computing Machinery, 2022. doi: 10.1145/3510466.3511274.

[133] **K. Meixner**, K. Feichtinger, R. Rabiser, and S. Biffl. Efficient Production Process Variability Exploration. In P. Arcaini, X. Devroey, and A. Fantechi, editors, *VaMoS '22: 16th International Working Conference on Variability Modelling of Software-Intensive Systems, Florence, Italy, February 23 - 25, 2022*, pages 14:1–14:9. ACM, 2022.

[134] **K. Meixner**, J. Musil, A. Lueder, D. Winkler, and S. Biffl. A coordination artifact for multi-disciplinary reuse in production systems engineering. In *27th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2022, Stuttgart, Germany, September 6-9, 2022*, volume 2022-September, pages 1–8. Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ETFA52439.2022.9921586.

[135] **K. Meixner**, F. Rinker, L. Waltersdorfer, A. Lueder, and S. Biffl. Organizing reuse for production systems engineering with capabilities and skills. *At-Automatisierungstechnik*, 71(2):116–127, 2023. doi: 10.1515/AUTO-2022-0120.

[136] **K. Meixner**, K. Feichtinger, H. S. Fadhlillah, S. Greiner, H. Marcher, R. Rabiser, and S. Biffl. Variability modeling of products, processes, and resources in cyber-physical production systems engineering. In *Proceedings of the 28th ACM International Systems and Software Product Line Conference - Volume A, SPLC 2024, Dommeldange, Luxembourg, September 2-6, 2024*, page 219. ACM, 2024. doi: 10.1145/3646548.3676547.

[137] **K. Meixner**, K. Feichtinger, S. Greiner, H. Marcher, H. S. Fadhlillah, R. Rabiser, and S. Biffl. Variability Modeling of Products, Processes, and Resources in Cyber-Physical Production Systems Engineering. *Journal of Systems and Software*, 211: 112007, 2024. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2024.112007.

[138] **K. Meixner**, K. Feichtinger, S. Greiner, and R. Rabiser. On Configuration Sequences in Feature Models. In T. Kehrer, M. Huchard, L. Teixeira, and C. Birchler, editors, *Proceedings of the 18th International Working Conference on Variability*

*Modelling of Software-Intensive Systems, VaMoS 2024, Bern, Switzerland, February 7-9, 2024*, pages 146–148. ACM, 2024. doi: 10.1145/3634713.3634730.

[139] **K. Meixner**, D. Hoffmann, S. Riedmann, P. Hünecke, and C. Binder. Rolling the Dice – Rethinking the RAMI 4.0 Perspectives. In *29th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2024, Padova, Italy, September 10-13, 2024 (in press)*, pages 1–8. IEEE, 2024. doi: 10.1109/ETFA61755.2024.10711108.

[140] **K. Meixner**, A. Köcher, F. Spitzer, V. Ashiwal, L. Sonnleithner, S. Schmitt, R. Froschauer, and A. Zoitl. Capabilities and Skills in Manufacturing: A Systematic Literature Study (planned). *OJIES*, pages 1–8, 2025.

[141] M. Mendonça and D. D. Cowan. Decision-making coordination and efficient reasoning techniques for feature-based configuration. *Science of Computer Programming*, 75(5):311–332, 2010.

[142] S. Mennicke, M. Lochau, J. Schroeter, and T. Winkelmann. Automated verification of feature model configuration processes based on workflow petri nets. In *SPLC*, pages 62–71. ACM, 2014.

[143] L. Monostori. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP*, 17:9–13, 2014.

[144] J. M. Neighbors. *Software construction using components*. PhD thesis, University of California, Irvine, 1980.

[145] D. Nešić and M. Nyberg. Multi-view modeling and automated analysis of product line variability in systems engineering. In *Proceedings of the 20th International Systems and Software Product Line Conference*, SPLC '16, page 287–296, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340502.

[146] P. Novak, J. Vyskocil, P. Kadera, L. Kathrein, **K. Meixner**, D. Winkler, and S. Biffl. Engineering roles and information modeling for industry 4.0 production system engineering. In *24th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2019, Zaragoza, Spain, September 10-13, 2019*, volume 2019-September, pages 1669–1672. Institute of Electrical and Electronics Engineers Inc., 2019. doi: 10.1109/ETFA.2019.8869141.

[147] M. Nyberg. Generating safety cases for large-scale industrial product lines, 09 2021. URL `https://splc2021.net/program/keynotes`. Keynote at 25th ACM International Systems and Software Product Line Conference.

[148] K. Paetzold. Product and systems engineering/ca* tool chains. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*, pages 27–62. Springer, 2017.

[149] J. Pfrommer, M. Schleipen, and J. Beyerer. PPRS: Production skills and their relation to product, process, and resource. In *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–4, 2013.

[150] J. Pfrommer, D. Stogl, K. Aleksandrov, S. Escaida Navarro, B. Hein, and J. Beyerer. Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems. *At-Automatisierungstechnik*, 63:790–800, 2015. doi: 10.1515/auto-2014-1157.

[151] Plattform Industrie 4.0 and ZVEI. Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01 Review). Standard, German BMWI, Nov. 2020. `https://bit.ly/37A002I`.

[152] K. Pohl, G. Böckle, and F. J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer Science & Business Media, 2005.

[153] A. Prock. Hybrid Human-Machine Ontology Verification: Identifying Common Errors in Ontologies by Itegrating Human Computation with Ontology Reasoners. Master's thesis, TU Wien, Wien, 2021.

[154] S. Profanter, A. Perzylo, M. Rickert, and A. Knoll. A Generic Plug Produce System Composed of Semantic OPC UA Skills. *IEEE Open Journal of the Industrial Electronics Society*, 2:128–141, 2021. ISSN 2644-1284. doi: 10.1109/OJIES.2021.3055461.

[155] C. Quinton, D. Romero, and L. Duchien. Cardinality-based feature models with constraints: A pragmatic approach. In *17th International Software Product Line Conference*, SPLC '13, page 162–166. ACM, 2013. ISBN 9781450319683.

[156] M. Raatikainen, J. Tiihonen, and T. Männistö. Software product lines and variability modeling: A tertiary study. *Journal of Systems and Software*, 149:485–510, 2019. ISSN 0164-1212.

[157] R. Rabiser and A. Zoitl. Towards Mastering Variability in Software-Intensive Cyber-Physical Production Systems. In *Proceedings of the 2nd International Conference on Industry 4.0 and Smart Manufacturing (ISM 2020), Virtual Event, Austria, 23-25 November 2020*, volume 180 of *Procedia Computer Science*, pages 50–59. Elsevier, 2020.

[158] R. Rabiser, P. Grünbacher, and M. Lehofer. A qualitative study on user guidance capabilities in product configuration tools. In *27th IEEE/ACM International Conference on Automated Software Engineering*, pages 110–119. ACM, 2012.

[159] R. Rabiser, K. Schmid, M. Becker, G. Botterweck, M. Galster, I. Groher, and D. Weyns. Industrial and Academic Software Product Line Research at SPLC: Perceptions of the Community. In *23rd International Systems and Software Product Line Conference*, pages 189–194, Paris, France, 2019. ACM.

[160] RAMI 4.0. Reference Architecture Model Industrie 4.0 (RAMI 4.0). Beuth Verlag, 2016.

[161] A. Rauber, B. Gößwein, C. M. Zwölf, C. Schubert, F. Wörister, J. Duncan, K. Flicker, K. Zettsu, **K. Meixner**, L. D. McIntosh, S. Pröll, T. Miksa, and M. A. Parsons. Precisely and Persistently Identifying and Citing Arbitrary Subsets of Dynamic Data. *Harvard Data Science Review*, 3(4), nov 18 2021. doi: 10.1162/99608f92.be565013. https://hdsr.mitpress.mit.edu/pub/si7wzxxa.

[162] M. O. Reiser and M. Weber. Multi-level feature trees: A pragmatic approach to managing highly complex product families. *Requirements Engineering*, 12(2):57–75, 2007.

[163] F. Rinker, L. Waltersdorfer, **K. Meixner**, and S. Biffl. Towards support of global views on common concepts employing local views. In *24th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2019, Zaragoza, Spain, September 10-13, 2019*, volume 2019-September, pages 1686–1689. Institute of Electrical and Electronics Engineers Inc., 2019. doi: 10.1109/ETFA.2019.8869239.

[164] F. Rinker, **K. Meixner**, L. Waltersdorfer, D. Winkler, A. Lueder, and S. Biffl. Towards efficient generation of a multi-domain engineering graph with common concepts. In *26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021*, volume 2021-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ETFA45728.2021.9613657.

[165] F. Rinker, L. Waltersdorfer, **K. Meixner**, D. Winkler, A. Lueder, and S. Biffl. Continuous integration in multi-view modeling: A model transformation pipeline architecture for production systems engineering. In S. Hammoudi, L. F. Pires, E. Seidewitz, and R. Soley, editors, *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2021, Online Streaming, February 8-10, 2021*, pages 286–293. SCITEPRESS, 2021. doi: 10.5220/0010309902860293.

[166] F. Rinker, S. Kropatschek, T. Steuer, **K. Meixner**, E. Kiesling, A. Lueder, D. Winkler, and S. Biffl. Efficient multi-view change management in agile production systems engineering. In J. Filipe, M. Smialek, A. Brodsky, and S. Hammoudi, editors, *Proceedings of the 24th International Conference on Enterprise Information Systems, ICEIS 2022, Online Streaming, April 25-27, 2022, Volume 2*, volume 2, pages 134–141. Science and Technology Publications, Lda, 2022. doi: 10.5220/0011074000003179.

[167] F. Rinker, **K. Meixner**, S. Kropatschek, E. Kiesling, and S. Biffl. Risk and engineering knowledge integration in cyber-physical production systems engineering. In G. M. Callicó, R. Hebig, and A. Wortmann, editors, *48th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2022, Maspalomas,*

*Gran Canaria, Spain. 31 August - 2 September 2022*, pages 338–345. Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/SEAA56994.2022.00060.

[168] F. Rinker, S. Kropatschek, T. Steuer, E. Kiesling, **K. Meixner**, L. Waltersdorfer, P. Sommer, A. Lueder, D. Winkler, and S. Biffl. Multi-view fmea re-validation: Efficient risk and engineering knowledge integration in agile production systems engineering. *Communications in Computer and Information Science*, 1708 CCIS: 60–83, 2023. doi: 10.1007/978-3-031-38821-7_4.

[169] F. Rinker, D. Vysoka, **K. Meixner**, D. Hoffmann, and S. Biffl. Organizing multi-domain change impact analysis in cyber-physical production systems engineering. In *28th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2023, Sinaia, Romania, September 12-15, 2023*, volume 2023-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ETFA54631.2023.10275684.

[170] F. Rinker, L. Waltersdorfer, **K. Meixner**, D. Winkler, A. Lueder, and S. Biffl. Traceable multi-view model integration: A transformation pipeline for agile production systems engineering. *SN Computer Science*, 4(2):205, 2023. doi: 10.1007/s42979-022-01572-5.

[171] F. Rinker, **K. Meixner**, R. Dogaru, and S. Biffl. Graph-Based Change Impact Visualization for Agile Cyber-Physical Production Systems Engineering. In *29th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2024, Padova, Italy, September 10-13, 2024*, pages 1–4. IEEE, 2024. doi: 10.1109/ETFA61755.2024.10710638.

[172] F. Rinker, **K. Meixner**, D. Vysoká, and S. Biffl. Multi-Domain Modeling for Change Management in Cyber-Physical Production Systems Engineering. In *29th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2024, Padova, Italy, September 10-13, 2024 (in press)*, pages 1–8. IEEE, 2024. doi: 10.1109/ETFA61755.2024.10710656.

[173] F. Rinker, D. Vysoká, **K. Meixner**, and S. Biffl. Survey of Practitioner Needs and Approaches for Multi-Domain Change Management in Cyber-Physical Production Systems Engineering. In *29th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2024, Padova, Italy, September 10-13, 2024*, pages 1–8. IEEE, 2024. doi: 10.1109/ETFA61755.2024.10711158.

[174] D. Rombach. Integrated software process and product lines. In *Software Process Workshop*, pages 83–90. Springer, 2005.

[175] M. Rosenmüller, N. Siegmund, T. Thüm, and G. Saake. Multi-dimensional variability modeling. In *Proc. of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '11, pages 11–20, New York, NY, USA, 2011. ACM.

[176] E. Rouillé, B. Combemale, O. Barais, D. Touzet, and J.-M. Jézéquel. Leveraging cvl to manage variability in software process lines. In *19th Asia-Pacific Software Engineering Conference*, volume 1, pages 148–157. IEEE, 2012.

[177] P. Runeson, M. Host, A. Rainer, and B. Regnell. *Case study research in software engineering: Guidelines and examples.* John Wiley & Sons, 2012.

[178] H. Röpke. *Entwicklung einer Methode zur Risikobeurteilung bei der Wiederverwendung von Entwurfselementen im Anlagenengineering (Development of a risk assessment method for the reuse of design elements in plant engineering).* phdthesis, Otto-von-Guericke University, 2019.

[179] S. A. Safdar, H. Lu, T. Yue, S. Ali, and K. Nie. A framework for automated multi-stage and multi-step product configuration of cyber-physical systems. 20(1): 211–265, 2021.

[180] M. Sarna, **K. Meixner**, S. Biffl, and A. Lueder. Reducing risk in industrial bin picking with pprs configuration and dependency management. In *26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021*, volume 2021-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ETFA45728.2021.9613618.

[181] M. Schleipen, A. Lüder, O. Sauer, H. Flatt, and J. Jasperneite. Requirements and concept for plug-and-work. *at-Automatisierungstechnik*, 63(10):801–820, 2015.

[182] K. Schmid, R. Rabiser, and P. Grünbacher. A comparison of decision modeling approaches in product lines. In *5th International Workshop on Variability Modelling of Software-Intensive Systems*, pages 119–126. ACM, 2011.

[183] G. Schuh, R. Anderl, J. Gausemeier, M. Ten Hompel, and W. Wahlster. Industrie 4.0 Maturity Index. Managing the Digital Transformation of Companies – UPDATE 2020. Technical report, acatech STUDY, 2020. URL `https://www.acatech.de/publikation/industrie-4-0-maturity-index-update-2020/`.

[184] F. Shull, J. Singer, and D. I. Sjøberg. *Guide to Advanced Empirical Software Engineering.* Springer, 2007.

[185] N. Siegmund, M. Rosenmüller, M. Kuhlemann, C. Kästner, S. Apel, and G. Saake. SPL Conqueror: Toward optimization of non-functional properties in software product lines. *Software Quality Journal*, 20(3-4):487–517, 2012.

[186] J. Simmonds, D. Perovich, M. C. Bastarrica, and L. Silvestre. A megamodel for software process line modeling and evolution. In *ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 406–415. IEEE, 2015.

[187] R. P. Smiraglia. *Domain Analysis*, pages 85–101. Springer International Publishing, Cham, 2014. ISBN 978-3-319-09357-4. doi: 10.1007/978-3-319-09357-4_10.

[188] L. Sonnleithner, A.-L. Hager, A. Zoitl, and **K. Meixner**. Iec 61499 skill-based distributed design pattern. In *28th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2023, Sinaia, Romania, September 12-15, 2023*, volume 2023-September, pages 1–8. Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ETFA54631.2023.10275380.

[189] H. Stachowiak. General model theory. *Springer*, 1973.

[190] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.

[191] M. Strembeck and U. Zdun. An approach for the systematic development of domain-specific languages. *Software: Practice and Experience*, 39(15):1253–1292, 2009.

[192] C. Sundermann, K. Feichtinger, D. Engelhardt, R. Rabiser, and T. Thüm. Yet another textual variability language? a community effort towards a unified language. In *Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume A*, SPLC '21, page 136–147, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384698. doi: 10.1145/3461001.3471145.

[193] C. Sundermann, K. Feichtinger, D. Engelhardt, R. Rabiser, and T. Thüm. Yet another textual variability language? a community effort towards a unified language. In *Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume A*, SPLC '21, page 136–147, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384698.

[194] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake. A classification and survey of analysis strategies for software product lines. *ACM Comput. Surv.*, 47(1), jun 2014. ISSN 0360-0300.

[195] T. Tolio, D. Ceglarek, H. A. ElMaraghy, A. Fischer, S. J. Hu, L. Laperrière, S. T. Newman, and J. Váncza. Species—co-evolution of products, processes and production systems. *CIRP Annals*, 59(2):672–693, 2010. ISSN 0007-8506.

[196] G. I. Trujillo-Tzanahua, U. Juárez-Martínez, A. A. Aguilar-Lasserre, and M. K. Cortés-Verdín. Multiple software product lines: applications and challenges. In J. Mejia, M. Muñoz, Á. Rocha, Y. Quiñonez, and J. Calvo-Manzano, editors, *Trends and Applications in Software Engineering*, pages 117–126, Cham, 2018. Springer International Publishing. ISBN 978-3-319-69341-5".

[197] F. van der Linden, K. Schmid, and E. Rommes. *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering*. Springer Berlin Heidelberg, 2007.

[198] VDI/VDE 2206. VDI/VDE 2206: Development of mechatronic and cyber-physical systems. Beuth Verlag, 1 part, 2021.

[199] VDI/VDE 3682. VDI/VDE 3682: Formalised Process Descriptions. Beuth Verlag 2 parts, 2015.

[200] VDI/VDE 3695. VDI/VDE 3695: Engineering of industrial plants. Evaluation and optimization of the engineering. Beuth Verlag, 5 parts, 2020.

[201] B. Vogel-Heuser and S. Biffl. Cross-discipline modeling and its contribution to automation. *Automatisierungstechnik*, 64(3):165–167, 2016.

[202] B. Vogel-Heuser, M. Böhm, F. Brodeck, K. Kugler, S. Maasen, D. Pantförder, M. Zou, J. Buchholz, H. Bauer, F. Brandl, and et al. Interdisciplinary engineering of cyber-physical production systems: highlighting the benefits of a combined interdisciplinary modelling approach on the basis of an industrial case. *Design Science*, 6:e5, 2020. doi: 10.1017/dsj.2020.2.

[203] D. M. Weiss and C. Lai. *Software Product-Line Engineering: A Family-Based Software Development Process*. Addison-Wesley, 1999.

[204] M. Weser, J. Bock, S. Schmitt, A. Perzylo, and K. Evers. An Ontology-based Metamodel for Capability Descriptions. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1679–1686, Sep. 2020. doi: 10.1109/ETFA46521.2020.9212104.

[205] R. J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[206] D. Winkler, **K. Meixner**, and S. Biffl. Towards flexible and automated testing in production systems engineering projects. In *23rd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2018, Torino, Italy, September 4-7, 2018*, volume 2018-September, pages 169–176. Institute of Electrical and Electronics Engineers Inc., 2018. doi: 10.1109/ETFA.2018.8502650.

[207] D. Winkler, L. Kathrein, **K. Meixner**, P. Staufer, M. Pauditz, and S. Biffl. Towards a hybrid process model approach in production systems engineering. *Communications in Computer and Information Science*, 1060:339–354, 2019. doi: 10.1007/978-3-030-28005-5_26.

[208] D. Winkler, **K. Meixner**, D. Lehner, and S. Biffl. Test reporting at a large-scale austrian logistics organization: Lessons learned and improvement. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11915 LNCS:53–69, 2019. doi: 10.1007/978-3-030-35333-9_4.

[209] D. Winkler, **K. Meixner**, and P. Novak. *Efficient and Flexible Test Automation in Production Systems Engineering*, pages 267–301. Springer International Publishing, 2019. doi: 10.1007/978-3-030-25312-7_9.

[210] D. Winkler, A. Lueder, **K. Meixner**, F. Rinker, and S. Biffl. Towards model consistency representations in a multi-disciplinary engineering network. In *25th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2020, Vienna, Austria, September 8-11, 2020*, volume 2020-September, pages 1413–1416. Institute of Electrical and Electronics Engineers Inc., 2020. doi: 10.1109/ETFA46521.2020.9211900.

[211] D. Winkler, P. Novak, **K. Meixner**, J. Vyskocil, F. Rinker, and S. Biffl. Product-process-resource asset networks as foundation for improving cpps engineering. In *26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021*, volume 2021-September, pages 1–4. Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ETFA45728.2021.9613253.

[212] D. Winkler, P. Novak, J. Vyskocil, **K. Meixner**, and S. Biffl. Industry 4.0 asset-based risk mitigation for production operation. In *17th IEEE International Conference on Automation Science and Engineering, CASE 2021, Lyon, France, August 23-27, 2021*, volume 2021-August, pages 278–285. IEEE Computer Society, 2021. doi: 10.1109/CASE49439.2021.9551419.

[213] C. Wohlin, A. Aurum, L. Angelis, L. Phillips, Y. Dittrich, T. Gorschek, H. Grahn, K. Henningsson, S. Kagstrom, G. Low, and others. The success factors powering industry-academia collaboration. *IEEE software*, 29(2):67–73, 2011.

[214] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering.* Springer Science & Business Media, 2012.

[215] P. Zimmermann, E. Axmann, B. Brandenbourger, K. Dorofeev, A. Mankowski, and P. Zanini. Skill-based Engineering and Control on Field-Device-Level with OPC UA. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 2019-September, pages 1101–1108, Sep. 2019. doi: 10.1109/ETFA.2019.8869473.