TU WIEN Informatics

Multi-Domain Change Impact Analysis for Agile Cyber-Physical Production Systems Engineering

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Diana Vysoká, BSc
Matrikelnummer 01633081

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl
Mitwirkung: Proj.Ass. Dipl.-Ing. Felix Rinker, BSc
Proj.Ass. Dipl.-Ing. Kristof Meixner, BSc

Wien, 30. Jänner 2024

Diana Vysoká

Stefan Biffl

TU WIEN Informatics

Multi-Domain Change Impact Analysis for Agile Cyber-Physical Production Systems Engineering

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Software Engineering & Internet Computing

by

Diana Vysoká, BSc
Registration Number 01633081

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl
Assistance: Proj.Ass. Dipl.-Ing. Felix Rinker, BSc
Proj.Ass. Dipl.-Ing. Kristof Meixner, BSc

Vienna, 30th January, 2024

Diana Vysoká | Stefan Biffl

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

Erklärung zur Verfassung der Arbeit

Diana Vysoká, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 30. Jänner 2024

Diana Vysoká

Acknowledgements

Although this thesis has only one author, many have contributed to the final result.

I would like to express my sincere gratitude to my thesis supervisor, Stefan Biffl, for the opportunity to work on this thesis within the Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle, for his guidance, prompt communication, and feedback throughout the journey.

I am deeply thankful to Felix Rinker for his assistance and dedication. Felix's mentorship, encouragement, constructive feedback, and countless hours of co-working sessions have been instrumental in shaping the direction and quality of this work. I thank Felix for the amazing opportunity to co-author a short conference paper for my first time and the valuable insights into academic research in this field.

I am also very thankful to Kristof Meixner, who introduced me to the topic of change impact analysis in CPPS engineering, provided constructive feedback and was very supportive throughout the journey.

Furthermore, I extend my appreciation to the industry partner who has provided assistance, shared their expertise, and collaborated with me throughout the evaluation of thesis results. Your support has been invaluable and deeply appreciated.

Lastly, I would like to acknowledge the support of my family and friends, whose encouragement, understanding, and patience have been my pillars of strength throughout this endeavor.

Thank you all for being part of this journey and your unwavering support.

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.

Kurzfassung

Die Fertigungs- und Produktionsindustrie steht vor dem Wandel zu intelligente Vernetzung von Maschinen und Abläufen mit Hilfe von Informations- und Kommunikationstechnologie bekannt als Industrie 4.0. Cyber-physische Produktionssysteme (CPPS) spielen eine zentrale Rolle bei diesem Wandel, da sie Flexibilität und Widerstandsfähigkeit bieten. Bei der Entwicklung von CPPS arbeiten Teams aus mehreren Domänen gleichzeitig und iterativ an einer Vielzahl von Assets, um CPPS, oder Teile davon, zu entwerfen und zu bauen. Der Schwerpunkt der Diplomarbeit liegt dabei auf der domänenübergreifenden CPPS Entwicklung, bei der Experten aus verschiedenen Domänen zusammenarbeiten.

Diese Diplomarbeit befasst sich mit den Herausforderungen der domänenübergreifenden Änderungsauswirkungsanalyse (M-CIA) der technischen Änderungen im Verlauf der CPPS Entwicklung. Zu den Herausforderungen einer solchen Analyse der Auswirkungen von Änderungen gehören der Austausch zwischen den Beteiligten aus unterschiedlichen Domänen, die hohe Komplexität der Integration sowie implizites und verstreutes Produktionswissen. Die Arbeit wendet die Design-Science-Methodik an und baut auf den agilen Change Management Workflows sowie bestehenden Software-Engineering-Ansätzen auf. Die Diplomarbeit 1) präsentiert die Ergebnisse einer Expertenbefragung, 2) identifiziert Herausforderungen von M-CIA in bestehenden Umgebungen 3) entwickelt die M-CIA Methode und ein unterstützendes Systemdesign.

Die Ergebnisse sind, mithilfe eines Prototypen, durch einer Machbarkeits- und einer Fallstudie validiert. Die Evaluierungsergebnisse der Methoden und des Systemdesign zeigen, dass der Lösungsansatz durchführbar ist. Durch die Fallstudie ist eine Effizienzsteigerung sowie die wahrgenommene Verbesserung ermittelt. Die vorgeschlagene M-CIA-Methode verwendet in der Softwareentwicklung etablierte Dev- und GitOps-Praktiken, um die domänenübergreifende Änderungsauswirkungsanalyse in CPPS zu erleichtern und zu automatisieren.

Die M-CIA-Methode und das Systemdesign ermöglicht Praktikern und Forschern die Koordination der domänenübergreifenden Änderungsauswirkungsanalyse in der CPPS-Umgebung zu verbessern. Dabei wird der domänenübergreifende Stakeholder-Austausch, die Zentralisierung des Produktionswissens und die Integration der domänenspezifischen Perspektiven auf ihr Produktionssystem erleichtert und somit kostspielige und späte Anpassungen verhindert.

Abstract

The manufacturing and production industry is on the verge of a transformation to intelligent networking of machines and processes with the help of information and communication technology, known as Industry 4.0. Cyber-Physical Production Systems (CPPSs) play a central role in this transformation as they offer flexibility and resilience. In CPPS engineering, teams from multiple domains work simultaneously and iteratively on a variety of assets to design and build CPPS, or parts of it. The focus of this thesis lies on multi-domain CPPS engineering, where experts from different domains work together.

This thesis addresses the challenges of Multi-domain Change Impact Analysis (M-CIA) of technical changes throughout the CPPS lifecycle. The challenges of such change impact analysis include the exchange between stakeholders from different domains, the high complexity of integration, and implicit and scattered production knowledge. The thesis applies the design science methodology and builds on agile change management workflows and existing software engineering approaches. The thesis 1) presents the results of an expert survey, 2) identifies challenges of M-CIA in existing environments 3) designs and validates the M-CIA method and a supporting system design.

The results are validated with a feasibility and a case study with the help of a system prototype. The evaluation results of the methods and the system design show that the solution approach is feasible. An increase in efficiency and perceived improvement are determined through a case study with an industry partner. The proposed M-CIA method utilizes established Dev- and GitOps practices in software development to facilitate and automate multi-domain change impact analysis in CPPS.

The M-CIA method and system design enable practitioners and researchers to improve the coordination of multi-domain change impact analysis in the CPPS environment. In doing so, it facilitates cross-domain stakeholder exchange, centralization of production knowledge, and integration of domain-specific perspectives on their production system, thus preventing costly and late adjustments.

Contents

CHAPTER 1

Introduction

This chapter introduces the topic of the thesis and motivates it by describing the current challenges in the CPPS engineering field. The chapter also introduces the envisioned solution approach and the structure of the thesis is presented. Finally, the chapter outlines expected contributions of the thesis.

1.1 Context & Motivation

The production and manufacturing industry face the *Fourth Industrial Revolution*, also called *Industry 4.0*. The trend towards automation, analytics, connectivity, and human-machine interaction is now part of smart factories, their technologies, and processes [Wu et al., 2019]. To enable this trend, factories utilize Cyber-Physical Systems (CPSs) to monitor the physical world by implementing the interaction between physical components and cyber components in distributed networks [Wu et al., 2019]. This thesis focuses on a sub-category of CPS applied in a production environment called CPPSs.

The main characteristics of a CPPSs compared to traditional production systems are its flexibility and customization [Meixner et al., 2021a]. These capabilities are also denoted as *self-X*, such as self-maintenance, self-repair or self-organization [Monostori, 2014].

While there is no formalized definition of CPPSs, Monostori [2014] defines CPPSs as systems that *"consist of autonomous and cooperative elements and sub-systems that are getting into connection with each other in situation-dependent ways, on and across all levels of production, from processes through machines up to production and logistics networks"*.

Production and manufacturing enterprises are facing a rapidly changing market [Yang et al., 2021]. To stay competitive, companies have to transform their assets. This work refers to products, processes, and resources that are used, produced, or applied in CPPS engineering as *assets*, based on [VDI/VDE 3682]. To conduct such a transformation,

changes in the existing assets have to be implemented. A change can be initiated due to various reasons, such as technical advancement, business objectives, or customer feedback. The changes are categorized as *manufacturing changes*, which describe reconfiguration or adaptation within the production systems and affect metrics such as production costs or duration [Koch et al., 2016]. Another type of change is *engineering changes*, which involves any alteration made to the system or its assets, which has already been put into operation [Koch et al., 2016]. This thesis focuses on engineering changes to the CPPSs.

Any change to a CPPS asset can have an impact on another CPPS asset. An engineering change can be either a *structural change* or a *value change*. Structural changes are changes made to the system, such as the addition or removal of an existing system part. Value changes are changes made to the system's assets, such as changes to value thresholds and system properties. It is crucial to conduct Change Impact Analysis (CIA) to understand the consequences of a requested change as well as enablers of a change before implementing the change request and putting the updated system to operation. Model-based engineering is a widely used approach to first represent the CPPS as a model, on which changes are implemented and validated before changing the real-world system [Heinrich et al., 2018].

CPPS engineering is a multi-domain effort. Meixner et al. [2021a] reports that up to 15 domains, such as mechanical, electrical, automation, or quality engineering, are involved in the engineering of a car assembly plant. This aspect adds complexity to change management and CIA processes.

1.2 Problem Description and Challenges

Due to its mentioned multi-domain nature and complexity, CPPS engineering faces several challenges that make the analysis of a change impact time-intensive, inefficient, and cumbersome. Software plays an important role in overcoming the challenges. However, the software engineering methods applied in CPPS engineering seem not to correspond to the newest software engineering methods, which have evolved tremendously in recent years [Feichtinger et al., 2022]. Feichtinger et al. [2022] conducted a systematic mapping of the challenges, based on workshops with industry representatives in CPPS engineering, and pointed out eight main challenges with multiple sub-categories.

The identified challenges by Feichtinger et al. [2022] are often conditioned by complexity, as CPPSs consist of a huge amount of heterogeneous subsystems (sensors, real-time control systems, process optimization systems), which makes it hard to carry out software integration or upgrade, ensure network interoperability and synchronization regarding real-time processes and applications, or security. These systems are long-living, which poses an additional challenge to the maintenance of the system, as requirements may change over time, and the systems have to adapt. Due to the multi-domain nature of the systems, multiple domain-specific development cycles have to be aligned, each having development cycles of different lengths. Additionally, each domain uses different processes and diverse tools that are poorly integrated across domains. Domain stakeholders have

varying knowledge of the system. Stakeholder interaction is typically based on the exchange of document-based artifacts, such as text files, spreadsheets, or tool exports.

The illustration of the context includes five domains (c.f. Figure 1.1), each domain has its specific goals and activities. Basic planners create comprehensive and optimized production plans and initial plans of the production systems that align with organizational goals and resource constraints. Mechanical engineers design and optimize physical components and systems within cyber-physical production, ensuring they meet performance, safety, and efficiency requirements. Electrical engineers design and implement electrical systems, components, and controls that integrate seamlessly with mechanical elements to achieve reliable and efficient production processes. Automation engineers design and implement automated solutions, including control systems and robotics, to enhance efficiency, reduce manual intervention, and ensure precision in cyber-physical production systems. Quality engineers establish and enforce quality standards, processes, and testing protocols within cyber-physical production systems to ensure the production of high-quality products with minimal defects or errors.

Figure 1.1 illustrates the multi-domain setting in CPPS engineering, in which stakeholders from different domains collaborate to engineer a CPPS and exchange data artifacts sequentially but also make changes that have to be communicated back to related stakeholders. In the course of the collaboration, these stakeholders face the challenges C1-C3 depicted in Figure 1.1, derived from the challenges identified by [Feichtinger et al., 2022].

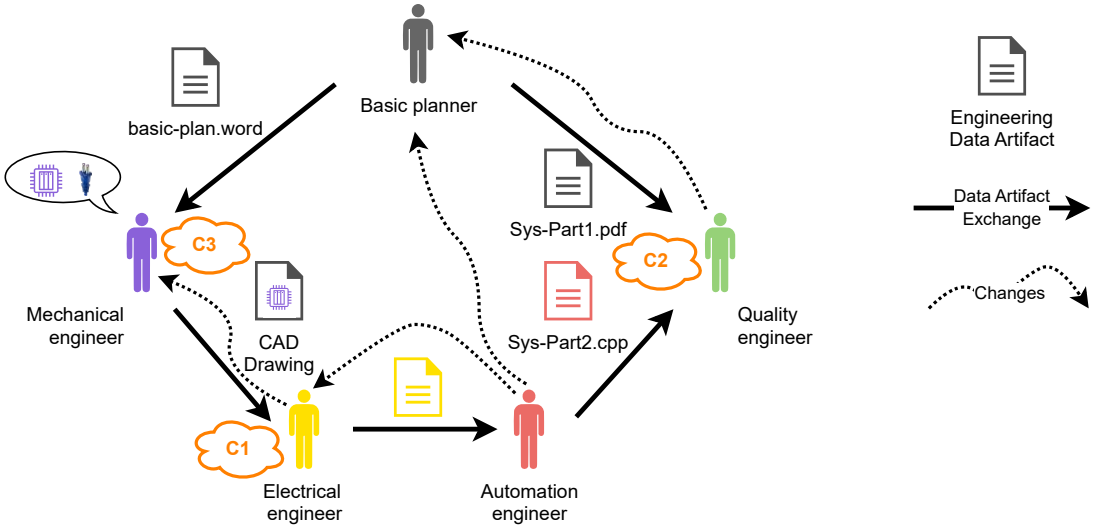

Figure 1.1: Traditional document-based exchange between CPPS engineering stakeholders with challenges (C1-C3) based on [Rinker, 2021]

C1. *Lack of holistic view of the system for multi-domain stakeholder exchange.* As shown in Figure 1.1 (c.f. C1), experts from domains such as electrical, mechanical,

and automation engineering are involved in the development and operation of a CPPS. Therefore, implementation of change requests to a CPPS requires input from stakeholders from multiple domains that often do not have a holistic overview of the system. This is due to different professional backgrounds, usage of domain-specific tools, and usage of different processes, which are poorly integrated across the domains. Additionally, the knowledge exchange necessary to facilitate change implementation and validation is often performed in a document-based fashion via e-mails, which can lead to data inconsistencies and is therefore risky, error-prone, and underlines the derived challenges [Meißner et al., 2021, Feichtinger et al., 2022]. The change coordination in the document-based approach is depicted with the dotted connector annotated as *Changes* in Figure 1.1.

C2. *High integration complexity hinders the facilitation of the common understanding.* To provide the necessary common understanding of the system, the integration of system artifacts is necessary. Such integration is a challenge because each domain has its local view of the system, and the views differ vertically (the abstraction level) or horizontally (focusing on specific sub-systems). Also, each domain has its specific file formats. As shown in Figure 1.1 (c.f. C2), a quality engineer receives a PDF file that includes documentation of a *system part 1* from a basic planner, which includes a high-level description. On the other hand, they also receive an automation script of *system part 2*. These two data artifacts are hard to integrate as they differ in vertical and horizontal dimensions.

C3. *Implicit and scattered production knowledge.* To facilitate a change, information from different sources and stakeholders has to be integrated to conduct CIA. This knowledge is often implicit, known only to selected engineers, not explicitly documented, and scattered across the CPPS ecosystem, which makes it difficult to implement and validate a change. As shown in Figure 1.1 (c.f. C3), a mechanical engineer prepares computer-aided drawings (CAD) of system parts. He knows the mechanical connections between system parts, e.g., controllers and plugs, and their physical properties (c.f. Figure 1.1). However, he only documents the physical properties of the controller because he assumes that the electrical engineer knows all the necessary details.

1.3 Overview and Structure of the Thesis

The thesis follows the Design Science methodology [Hevner, 2007] to design and validate the solution to the problem. We collaborate with industry partners of the Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI) and the members of the research group at TU Wien who research this field.

First, an elicitation of the state-of-the-art multi-domain CIA practice is conducted with an expert survey via a questionnaire. The thesis also consults the related work in the field of change management and CIA. Additionally, the thesis proposes the M-CIA method for

multi-domain CIA in CPPS engineering and a corresponding system design. The system design is implemented via a prototype, which facilitates the evaluation of the solution approach with feasibility and case study.

Chapter 2 introduces background on 1) multi-domain modeling for CPPS, 2) the foundation of the thesis which is the *agile Multi-view Change Management Workflow* [Rinker et al., 2022], 3) previously published research agenda, that was defined in the course of the thesis project, and guides the thesis [Rinker et al., 2023a], 4) the state-of-the-art collaboration tools for document-based exchange, and 5) selected fundamentals of graph theory.

Chapter 3 discusses related work in the field of CPPS engineering, knowledge representation in CPPS engineering, CIA in software engineering, change management approaches in CPPS engineering and Dev- and GitOps practice for change facilitation in the context of software and CPPS engineering.

Chapter 4 presents the research methodology in detail. First, the chapter introduces the research questions, how the thesis aims to address them, what their expected results are, and how to evaluate the results. Then, the chapter introduces *Design science* and the Information Systems Research Framework adapted to the topic of the thesis. Finally, the chapter explains how the activities from the M-CIA Framework will be carried out to answer the research questions and maps them to Design Science cycles.

Chapter 5 introduces the traditional document-based approach for multi-domain coordination and the illustrative use case from the car manufacturing industry. Finally, the chapter presents the minimal engineering and change management process for evaluation of the proposed M-CIA approach compared to the traditional approaches.

Chapter 6 introduces the solution approach represented by an expert survey and the M-CIA method to address the research questions. First, the chapter describes the expert survey and its preliminary results and key learnings. Then, the chapter introduces requirements for multi-domain CIA, the M-CIA method, and the proposed system design. Finally, the chapter introduces the system prototype and discusses its feasibility.

Chapter 7 reports on the evaluation of the M-CIA method and the system prototype evaluated with a case study. The case study is conducted with a selected production company. The case study evaluation focuses on the efficiency and perceived improvement of the proposed method and the system design compared to the traditional document-based approach. Efficiency is measured with estimated execution time metrics based on the *Key-Stroke Level Model* method. The perceived improvement of the method is evaluated using a 5-point *Likert scale* by a basic planner of the industry partner, who compared the proposed approach to the traditional document-based approaches relatively.

Chapter 8 discusses the results of each of the research questions, outlines the limitations of the solution approach, and lists threats to the validity of the work.

Finally, Chapter 9 concludes the thesis, discusses the uptake of research results in scientific communities and practice, and outlines future work topics.

1.4 Contributions

To convey the contributions of the thesis, this section describes the main deliverables:

Requirements analysis: Investigation of the current practice of multi-domain CIA in the industry and academia and derivation of the requirements for a multi-domain CIA method. The thesis presents the results and key learnings of the expert survey conducted using a questionnaire. Additionally, the thesis consults the requirements for efficient change management processes in CPPSs engineering in related work.

Knowledge modeling: Creation of the knowledge representation, based on the related work in regards to multi-view domain modeling, to formally describe a CPPS and its domain concepts to generate a holistic system model later. Based on the holistic system model, a knowledge graph is generated that depicts a production system. Based on the knowledge graph, the changes to the system's assets are identified, and the impact is analyzed.

Method design and validation: Design and validation of the M-CIA method, based on the requirements, to explore change dependencies and to coordinate the multi-domain CIA. The method will be validated in terms of feasibility on an illustrative use case *Fasten Screw and Measure* from the car manufacturing industry and evaluated on an evaluation use case *Fertilizer Mixing* from the fertilizer production industry to represent discrete and batch manufacturing in the experiments and to evaluate the efficiency of the solution approach.

System design and validation: Validation of the system design by implementing the proposed system architecture in a prototype that facilitates the M-CIA method execution and automates it.

These deliverables contribute to the related research communities as follows: The expert survey contributes the identified gaps in multi-domain CIA to the information and production systems research communities. The identified gaps can be seen as a potential research opportunity and a representation of the state of the practice from the practitioner's perspective.

Practitioners (c.f. Figure 1.1) can focus on addressing the challenges C1-C3 to build a holistic view of the system. For that, they are provided with an efficient method that guides them through formally modeling their assets and defining the dependencies between them to enable multi-domain CIA. The proposed system design supports them in identifying changes by automating the impact analysis coordination and providing guidance for the impact analysis throughout the engineering and change process.

Additionally, this thesis contributes to information and production systems research communities with a foundation for advanced approaches to multi-domain CIA, as the proposed approach provides means to gather validated and accurate production system

data (models). These models can be leveraged to design AI-driven and ontology-based methods to multi-domain CIA, as well as to facilitate digital transformation by deriving digital twins.

Finally, this thesis provides insights into the batch production process use case. Previous related work focuses on discrete production processes.

To uptake the contributions and benefit from them, the limitations, threats to validity, and future work of the thesis should be considered.

CHAPTER 2

Background

This chapter introduces the relevant concepts for the thesis. First, relevant modeling methods for multi-domain setup in CPPS engineering are presented as a prerequisite to eliciting use case data for method design and evaluation. The chapter also introduces the *agile Multi-view Change Management Workflow* as it is the foundation of our solution approach [Rinker et al., 2022]. Then, the chapter describes the M-CIA Framework that was published as a foundation and frame for the research agenda for this thesis [Rinker et al., 2023a], co-authored by the thesis' author. Finally, the basics of graph theory and knowledge graphs are introduced to provide an understanding of the system design and the prototype.

2.1 Multi-domain Modeling Method for CPPS

To analyze data for any purpose, e.g., for CIA, the data have to be modeled in a structured way using an appropriate method. Domain modeling is not part of the contribution of this thesis. Therefore, the solutions approach utilizes existing methods from the related work. Concretely, the solution approach utilizes the Multi-Domain Modeling for CPPS (MDM-CPPS) from the previous publication [Rinker et al., 2024] which is based on the MDM-CPPS Integrated Development Environment (IDE)[1].

This section presents the MDM-CPPS method as a prerequisite for the actual solution approach that will be developed as a contribution to the thesis.

The MDM-CPPS method combines domain-specific concepts [Rinker et al., 2023b] into an integrated engineering model, and is based on the *GitOps* practice known from software engineering. In this practice, the single source of truth is the configuration files, source code, and other artifacts *committed* to a Git-based repository.

[1] MDM-CPPS IDE: `https://marketplace.visualstudio.com/items?itemName=ModelIEE.mdmcpps-ide`

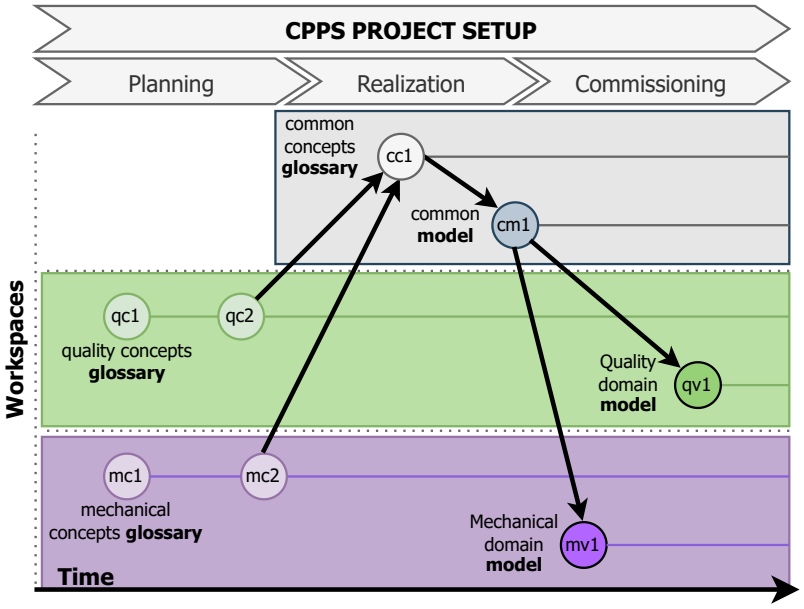

Figure 2.1: Illustrative project setup phase of the MDM-CPPS method, as proposed by Rinker et al. [2024].

The proposed Multi-Domain Modeling (MDM) method for CPPS engineering consists of two phases. The *project setup* phase is conducted at the beginning of an engineering project and facilitates the modeling of the relevant system knowledge. The *engineering and change management* phase represents the engineering process with changes and will be introduced as part of the solution approach.

In the *project setup* phase, the CPPS engineers define their local concepts in a distinct domain-specific workspace and negotiate a holistic view of the system's common concepts collaboratively [Rinker et al., 2024].

2.1.1 MDM-CPPS Project Setup

Figure 2.1 shows the *MDM-CPPS project setup* phase and its three activities, *Planning*, *Realization*, and *Commissioning*.

Planning. During the *Planning* activity, engineers establish *local workspaces* for individual domains within the project. These domain-specific workspaces function as specialized environments for the development of concepts by domain-specific teams. Within these designated spaces, team members formulate and collaborate on domain-specific Concept Glossaries (CGs) customized to depict the CPPS from their domains's perspective [Rinker et al., 2019].

In Figure 2.1, we depict CGs for both *mechanical* and *quality* domains, denoted as *mc1* in the mechanical workspace and *qc1* in the quality workspace, within the context of an

illustrative CPPS engineering project setup. As an illustrative example, suppose that the mechanical domain would develop a glossary with the resource *Screwdriver* and its corresponding mechanical attributes. These initially defined concepts and attributes can evolve and might therefore be subject to refinement through domain-specific discussions, as indicated by labels *mc2* and *qc2* in Figure 2.1, within the *Planning* activity.

The Concept Glossary (CG) file has an extension *.cg* and is stored in the domain-specific *local workspace*. Concepts in the concept glossaries have an ID and two fields: *name* and *attributes*. Attributes are also defined in the domain CG having a *name*, *defaultValue*, *type*, and *unit* field. An exemplary CG definition of the mechanical domain is shown Listing 1. The glossary starts with the glossary id definition *MechanicalConcepts*, name, and version. Then, we define the concept *c_m_electric_screwdriver* with its name *Electric Screwdriver* and the attribute *torque*.

```
ID MechanicalConcepts {
  name: "Mechanical Domain Concepts Glossary"
  version: 1.0.0
}
Attribute torque {
  name: "torque"
  defaultValue: 0.0
  type: "Number"
  unit: "Nm"
}
Concept c_m_electric_screwdriver {
  name: "Electric Screwdriver"
  attributes: torque
}
```

Listing 1: Illustrative *concept* definition *Electric Screwdriver* with attribute *torque* for the mechanical domain as reported in [Rinker et al., 2024].

Realization. In the *Realization* activity, a lead engineer sets up a *common workspace* that will contain the holistic view of the CPPS negotiated and approved by engineering teams from all domains. In this common workspace, during the negotiation process, the engineering teams agree on and describe Common Concepts (CCs) for the CPPS in Common Concepts Glossary (CCG) (Figure 2.1 label *cc1* in the common workspace) based on their local concepts [Rinker et al., 2019].

These domain concepts and the overarching common concepts collectively form a taxonomy that can be well represented in a knowledge graph [Rinker et al., 2021]. For instance, consensus might be reached on a taxonomy for screwdrivers, distinguishing between electric and pneumatic types and specifying attributes relevant to various domains. Furthermore, relationships can be defined at the domain concept and common concept levels. A relationship, in this context, signifies a dependency between attributes of two interrelated objects.

Subsequently, a Git-based workflow [Halilaj et al., 2016] can be employed to map the previously defined local concepts, extracted from their respective domain glossaries, onto the common concepts within the CCG [Rinker et al., 2019].

CCs are defined in a CCG file with the extension *.ccg* and stored in the *common workspace*. The CCG is specified by a unique *ID*, *name*, and *version* field. A CC has a unique ID and the two fields: *name*, and a list of concepts, *inhabits*, which the common concept inhabits. These can originate from various domain concept glossaries. Listing 2 shows the CC with ID *cc_electric_screwdriver* which inhabits the mechanical concept *c_m_electric_screwdriver* and the electrical concept *c_e_electric_screwdriver*.

```
ID CommonConceptGlossary {
  name: "Common Concept Glossary"
  version: 1.0.0
}
CommonConcept cc_electric_screwdriver {
  name: "Electric Screwdriver"
  inhabits:
    MechanicalConcepts.c_m_electric_screwdriver,
    ElectricalConcepts.c_e_electric_screwdriver
}
```

Listing 2: Illustrative CC definition of an *Electric Screwdriver*, which inhabits *concepts* from the mechanical and electrical domain, as reported in our previous work [Rinker et al., 2024].

Finally, the lead engineer, supported by the domain teams, defines a *common model* of the CPPS (label *cm1* in the common concepts workspace) that can also be iteratively refined. The *common model* contains the assets of the CPPS, which can be products, processes, or resources. Each of the assets has to represent a common concept, which again inhabits a corresponding domain concept.

To illustrate this relation, consider Figure 2.1 with the following example: Mechanical engineers define a screwdriver as a concept in their mechanical *Concept Glossary*. Electrical engineers define an electric screwdriver as a concept in their electrical *Concept Glossary*. During the negotiation process, the engineers agree that these two domain concepts are the same (although for each domain, different properties are relevant) and can, therefore, represent a common concept called *screwdriver*. This common concept is then documented in the *Common Concepts Glossary* [Rinker et al., 2019]. Finally, the lead engineer creates a *common model*, in which he defines a resource *Screwdriver*. The common workspace now contains the CCG and the unified common model based on a Single Underlying Model (SUM) Tunjic and Atkinson [2015] that represents all relevant assets with their corresponding common concept references. This collectively agreed-upon SUM serves as the foundation for the engineering project, ensuring uniformity and coherence across diverse domains.

Common model file is located in the *common workspace* with the file extension *.ppr*. The

common model has a header with a unique *ID*, *name*, and *version*. *Product*, *Process* or *Resource* assets are defined with a unique *ID* and have at least two fields: name and field *represents*, which refers to a CC.

Listing 3 shows a resource with ID *electric_screwdriver* that represents the CC *cc_electric_screwdriver* and thus also instantiates the domain-specific attributes mechanical *torque* and electrical *power_consumption*.

```
ID PositionScrewDashboard_Model {
  name: "Model Fasten Screw and Measure Use Case"
  version: 1.1.0
}
Resource electric_screwdriver {
  name: "Electric Screwdriver"
  represents: CommonConceptGlossary.cc_electric_screwdriver
  children: bit
  parents: robot
  requires: drive
  ElectricalConcepts.power_consumption: 0.0
  MechanicalConcepts.torque: 0.0
}
```

Listing 3: Product-Process-Resource (PPR) *common model* definition *Electric Screwdriver* representing the CC *Electric Screwdriver* with attributes *power_consumption* and *torque* as reported in our previous work [Rinker et al., 2024]

Commissioning. In the *Commissioning* activity, we finally derive the domain-specific models (Figure 2.1 labels *qv1* and *mv1* in the local workspaces) of the CPPS based on the validated and finalized common model, common concepts, and corresponding domain concepts. The domain-specific models are a projection of the system common model to a domain and only contain the assets which are represented by domain concepts. These domain-specific models of the CPPS allow teams to focus on domain-specific concerns later in the multi-domain *engineering and change management* phase while aligning with the overarching project structure.

In the multi-domain *engineering and change management* phase, engineers make changes to the assets in the domain-specific models generated in *Commissioning* activity, and merging them back to the common model requires a coordinated change management process supported by an appropriate system design, e.g. Traceable Multi-view Model Transformation (TMvMT) as proposed by Rinker et al. [2023b].

2.1.2 Meta Object Factory Architecture for MDM-CPPS Method

The definition of the domain concepts, common concepts, the corresponding PPR assets and relationships between them, is facilitated with the MDM-CPPS Domain-specific

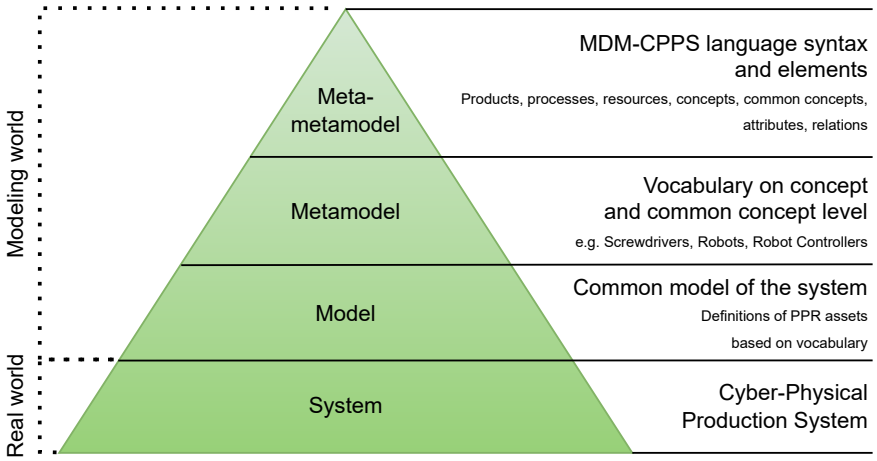

Figure 2.2: Meta Object Factory (MOF) Architecture pyramid depicting the modeling layers of the MDM-CPPS method, based on Orłowski et al. [2016].

Language (DSL)[2]. This DSL is a domain-specific language that provides the syntax and language elements to describe a model of a CPPS formally. The MDM-CPPS DSL is based on the PPR DSL proposed by Meixner et al. [2021b] and extended for the concepts and common concepts language elements as proposed by [Rinker et al., 2023b].

In Figure 2.2, we depict the modeling layers using the MOF Architecture pyramid [Orłowski et al., 2016]. The meta-metamodel layer is the most general description of the modeling world, concretely the MDM-CPPS language syntax and its elements. The metamodel layer is the vocabulary of the domain and common concepts that are later used to model the actual CPPS. The model layer refers to the models of the system described using the modeling languages specified in meta- and meta-metamodel. Finally, the system layer represents the real CPPS in the real world.

2.2 Agile Multi-view Change Management Workflow

Rinker et al. [2022] provide the foundation of the research topic, which is the agile Multi-view Change Management (MvCM) workflow based on the Git Workflow[3] and pull requests for changes of CPPS assets.

This change management workflow was introduced as an alternative to the document-based uncoordinated change management, which was illustrated in Figure 1.1. Figure 2.3 illustrates the steps of the workflow in a scenario of a property change: 1) local

[2]MDM-CPPS IDE: `https://marketplace.visualstudio.com/items?itemName=ModelIEE.mdmcpps-ide`

[3]Git Workflow: `https://www.atlassian.com/git/tutorials/comparing-workflows`

preparation, 2) multidisciplinary change analysis, 3) multidisciplinary examination, 4) local rework, and 5) common integration. The thesis will focus on step 2) *multidisciplinary change analysis* and facilitating the decision "has impact?" in Figure 2.3.

The initial work neither specifies who is involved in the CIA and how to establish this knowledge nor defines how the review process is coordinated nor how the stakeholders are notified. The thesis will address this gap.

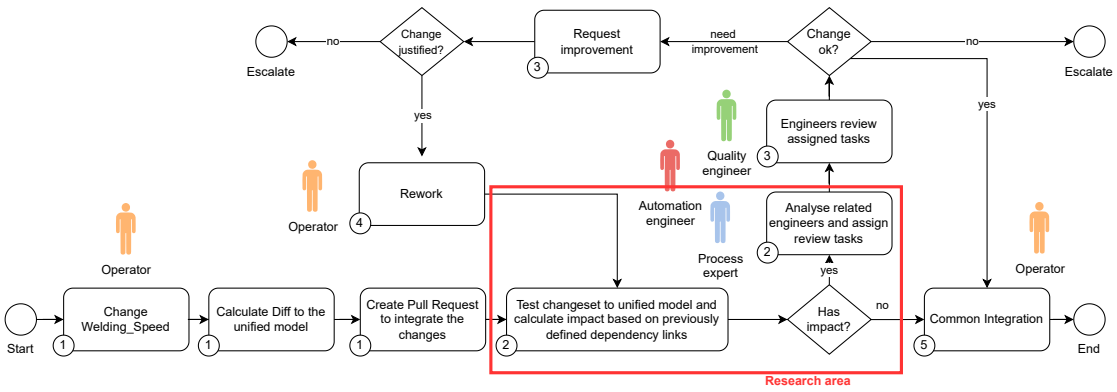

Figure 2.3: Multi-View Change Management Workflow based on Git Workflow as proposed by Rinker et al. [2022].

Within the *local preparation* step, an operator changes a property of an asset in a CPPS. Then, the difference to the unified model (also known as *common model* in the previous section) is recognized. Finally, the operator creates a pull request to integrate the change set into the unified model.

In the *multidisciplinary change analysis* step, the change set is tested with the unified model to check the system's validity. The impact is identified based on previously defined semantic links (relationships between system objects). If the impact analysis finds an impact of a change, a review of the change is triggered and assigned to a related engineer.

Otherwise, the workflow navigates to the last step, *common integration*, and the change set is integrated into the unified model.

In the *multidisciplinary examination* step, the related engineers assigned to a review task assess the change implementation and whether it is correct or an improvement is necessary. If an improvement is necessary, the reviewer request a rework.

Finally, in the *local rework* phase, an engineer assigned to perform the rework either finalizes the task, and the process proceeds to step 2 or escalates the issue to the project manager.

2.3 Multi-domain Change Impact Analysis Framework

Throughout the thesis project, we proposed the Multi-domain Change Impact Analysis (M-CIA) Framework based on [VDI/VDE 3695] and [Meixner et al., 2023] to outline the research plan and agenda on which this thesis is based.

The [VDI/VDE 3695] guideline describes best practices for engineering industrial plants. We designed the M-CIA Framework to facilitate tackling the challenges from Section 1.2 to design and evaluate a multi-domain CIA method for a CPPS organization. As part of the submission of our work at the *Emerging Technologies and Factory Automation 2023* conference, which is an established conference for industry practitioners and researchers [Wortmann et al., 2020], we gained valuable feedback and a lot of interest from the conference and the industry [Rinker et al., 2023a]. With this publication, we validated the soundness of the research approach in the CPPS research community.

Figure 2.4 shows the M-CIA Framework. The framework's main contribution is a structured approach to creating a *Common Knowledge Base* on the organizational level, which is the result of *project-independent activities (PIAs)* (the upper light-green lane). A *Common Knowledge Base* is a collection of reusable artifacts that are later used as input for future projects in the project-dependent activities (PDAs) (the lower light-green lane). The reusable artifacts are produced during the PIAs such as domain and system analysis, process and artifact preparation, knowledge modeling, and knowledge validation.

Below, each activity of the M-CIA Framework is briefly described, based on our publication [Rinker et al., 2023a].

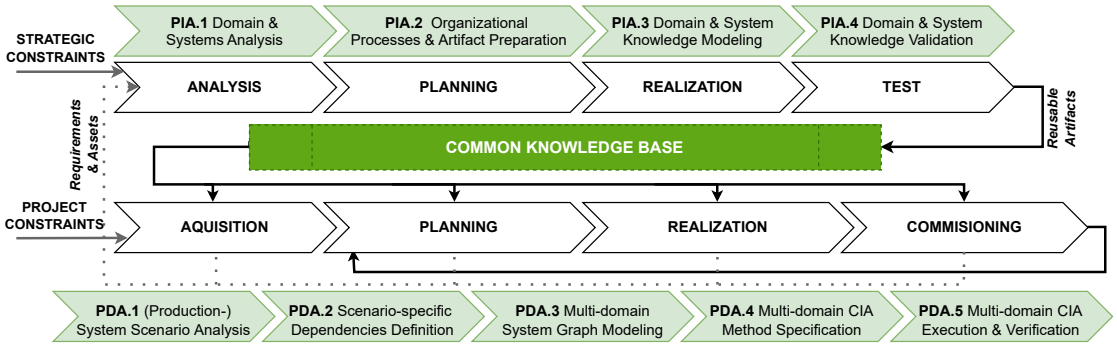

Figure 2.4: M-CIA Framework for conducting a CIA investigation project, based on the VDI/VDE 3695 from our publication [Rinker et al., 2023a].

2.3.1 Project-independent Activities

The project-independent activities (PIAs) (cf. Figure 2.4) are activities carried out at the beginning of an initiative to establish a common understanding of the domains, their inter-dependencies, and CPPSs within an organization. These activities build the

foundation, which we refer to as *Common Knowledge Base*, for further CIA investigation in the organization.

Strategic Constraints. Each organization navigates through a set of strategic constraints that define the organization's goals, corresponding activities, and requirements. Strategic constraints encompass factors such as government and domain regulations and security and safety requirements. These constraints influence the PIAs and implicitly influence PDAs through the reusable artifacts in the *Common Knowledge Base*.

PIA.1 – Domain & Systems Analysis. This activity matches the *Analysis* activity of the VDI/VDE 3695 (cf. Figure 2.4). *Activity:* The domain and systems analysis involves an exploration of techniques and tools associated with CIA from existing literature, assessing their relevance to multi-domain environment, and considering their potential adaptation and application within the selected environment. *Output:* The analysis results yield an informative summary of the diverse techniques and tools available for CIA across different fields.

PIA.2 – Organizational Process & Artifact Preparation. This activity matches the *Planning* activity of the VDI/VDE 3695 (cf. Figure 2.4). *Activity.* The organizational processes and artifacts should be prepared after the domain and system analysis. The activity describes the organization and system environment (stakeholders, roles, services, tools, and domain concepts), communication flows (e.g., chain of responsibility), and business and change management processes. *Output.* The activity results in artifacts, such as the documentation of system dependencies/interdependencies and the definition of domain concepts. Definition of domain concepts could follow [Rinker et al., 2019].

PIA.3 – Domain and System Knowledge Modeling. This activity matches the *Realization* activity of the VDI/VDE 3695 (cf. Figure 2.4). *Activity.* Based on the collected knowledge in the previous steps, the organization creates CCs to build a taxonomy based on artifacts and domain concepts from PIA.2. The definition process of CCs could follow [Rinker et al., 2019]. Additional project-independent reusable assets based on the taxonomy can be created. *Output.* The knowledge modeling activity contributes to the *Common Knowledge Base* with knowledge representation and reusable assets. The knowledge representation of the various concepts in the CPPSs, as well as the reusable assets, can later be used as the basis for PDAs.

PIA.4 – Domain and System Knowledge Validation. This activity matches the *Test* activity of the VDI/VDE 3695 (cf. Figure 2.4). *Activity.* After the reusable assets were created in the previous steps, the assets have to be validated to ensure consistency and correctness. In this activity, the organization coordinates a cross-domain validation exchange with domain representatives (e.g., based on chain of responsibility from PIA.2) to validate the outputs of PIA.3 and foster awareness of dependencies in the organization. *Output.* The output is a validated contribution to the *Common Knowledge Base* of the

organization (see Figure 2.4) in the form of process and method descriptions, modeled knowledge, information about domains, artifacts, and potential dependencies, that can be reused in future projects. The *Common Knowledge Base* is then used as an input for the project-dependent activities (PDAs).

2.3.2 Project-dependent Activities

Given the validated reusable artifacts that describe various aspects of the CPPS organization in *Common Knowledge Base*, an organization is enabled to start PDAs (cf. Figure 2.4). PDAs are activities carried out to design and validate new methods for CIA. Below, we briefly introduce the PDAs as we proposed in our previous publication [Rinker et al., 2023a].

Project Constraints. Every project is defined within the organization's context, tools used, resource and time constraints for project delivery, and quality requirements. These project constraints are input to the PDAs, as they must be considered in every project step. These constraints are influenced by *Strategic Constraints*.

PDA.1 – (Production-)System Scenario Analysis. This activity matches the *Acquisition* activity of the VDI/VDE 3695 (cf. Figure 2.4) and its primary input are *Project Constraints* and the reusable artifact from *Common Knowledge Base*. *Activity.* In this activity, scenarios have to be elicited, at least one illustrative use case as a basis for a CIA investigation project, as well as one evaluation use case for the method evaluation. *Output.* Possible outputs of the (Production-)System Scenario Analysis: Use case description, process diagrams, analysis of the use-case-relevant stakeholders.

PDA.2 – Scenario-Specific Dependencies Definition. This activity matches the *Planning* activity of the VDI/VDE 3695 (cf. Figure 2.4). *Activity:* As the next step, it is necessary to identify and understand scenario-specific dependencies between assets and domains and to develop knowledge representation of the CPPS. *Output.* The proposed output of this activity is a knowledge representation of assets and domain dependencies.

PDA.3 – Multi-domain system graph modeling. This activity matches the *Realization* activity of the VDI/VDE 3695 (cf. Figure 2.4). *Activity.* At this step of the framework, there is expected to be enough information and knowledge to develop machine-executable artifacts, such as knowledge graphs, that represent scenario-specific constraints, stakeholders/domains, processes, and dependencies. These artifacts are later executed as part of the newly designed method. *Output.* The expected output of this activity is machine-executable artifacts.

PDA.4 – Multi-domain Change Impact Analysis (CIA) method specification. This activity matches the *Realization* activity of the VDI/VDE 3695 (cf. Figure 2.4). *Input.* This activity uses the *Common Knowledge Base* and machine-executable artifacts

from the previous PDA.3. *Activity.* In this activity, a (manual or computer-supported) method to conduct the steps of the CIA on the specific use case should be designed. A method in this context is a collection of steps to analyze the impact of change and related cross-domain stakeholders. *Output.* The expected output of this activity is a method to conduct CIA on a given illustrative use case.

PDA.5 – Multi-domain CIA Execution and Verification. This activity matches the *Commissioning* activity of the [VDI/VDE 3695] (cf. Figure 2.4). *Input.* This activity uses the evaluation use case and the newly designed method from the previous PDA.4. *Activity.* Execute the method from PDA.4 on evaluation use cases to gain feedback and validate the method. As argued in [Rinker et al., 2023a], the M-CIA Framework extends the VDI/VDE 3695 procedure model so that it is possible to iteratively implement feedback gathered from the execution and verification by switching from *Commissioning* to *Planning* activities. *Output.* The newly developed and validated method for CIA in multi-domain engineering organizations, as well as relevant artifacts created in the project-dependent activities, can be used as input for a new iteration of the PIAs. Afterwards, these new artifacts can be introduced to the *Common Knowledge Base*.

The M-CIA Framework will guide the thesis project and provide a sound research approach, approved by the research community [Rinker et al., 2023a]. We will follow the framework steps throughout the thesis project to answer the research questions. Chapter 4 instantiates the framework in the thesis project's context.

2.4 State-of-the-art Collaboration Tools

This section introduces state-of-the-art collaboration tools, which are later used as a benchmark for the evaluation.

The proposed M-CIA method and system design are compared to traditional document-based approaches in the evaluation. These traditional approaches are realized through state-of-the-art collaboration tools. The traditional approaches are realized using the widely known and used *Microsoft 365* family of client software, concretely *Microsoft Outlook*, *Microsoft SharePoint*, and *Microsoft Teams*.

Microsoft Outlook is a personal information manager software, mostly used for email and calendar capabilities. Figure 2.5 shows the email module of *Microsoft Outlook*, referred to as the "main page of Outlook" later in the evaluation. The right side of the screenshot depicts the user interface that facilitates email writing. The user navigates to this interface via the blue button "New mail." Normally, a selected email from the list is opened in the detail view on the right side of the screenshot instead (currently, all are covered with the gray rectangle for privacy reasons).

Microsoft SharePoint is a web-based collaborative platform integrating with *Microsoft 365*. It is mainly used as a document management system with integrated storage.

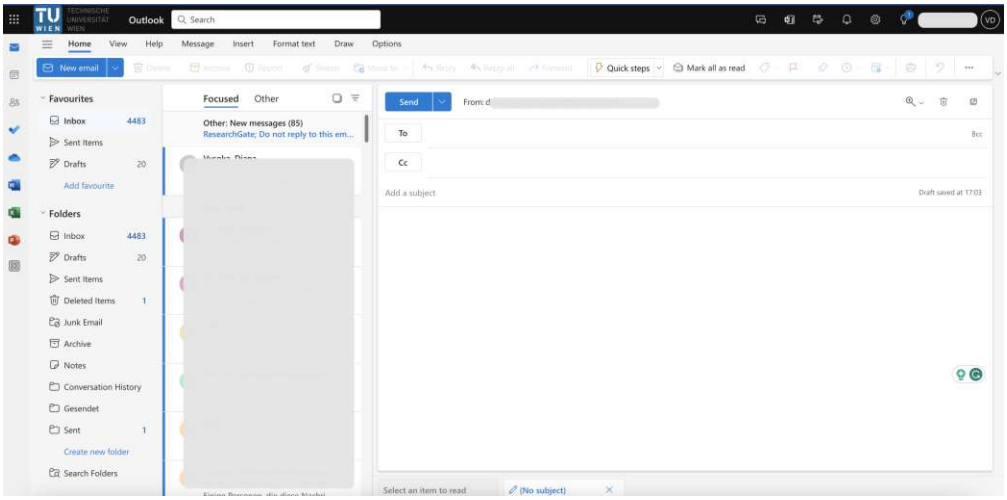

Figure 2.5: Outlook main page after the blue button "New mail" was clicked.

In *SharePoint*, *Excel*, *PowerPoint*, or *Word*, documents can be managed online and collaboratively. Figure 2.6 shows the SharePoint site created for demonstration purposes. On the left upper side, the site's name and logo are shown. On the right side of the screenshot, the folder structure is shown - one folder for the data artifacts and one folder for documenting change requests.

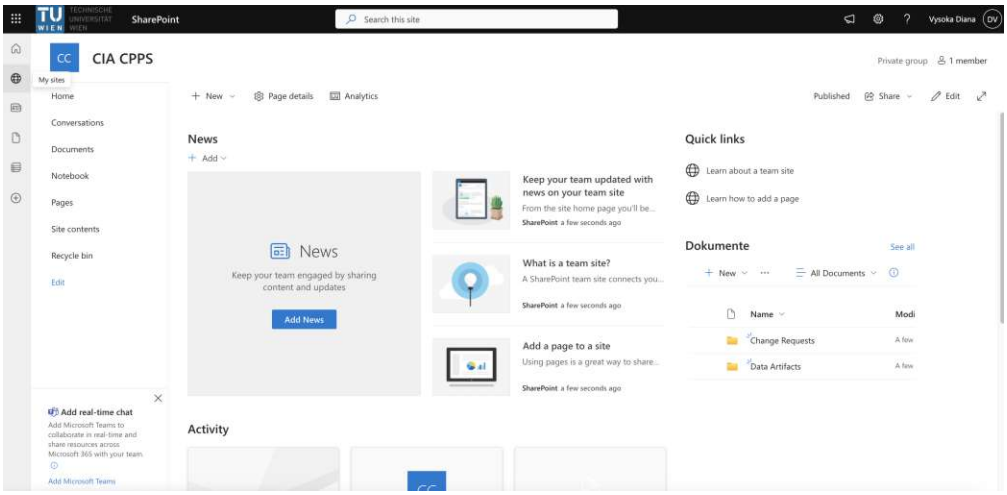

Figure 2.6: Site for the *CIA CPPS* project.

Microsoft Teams is a communication platform with chatting functionality and integration with SharePoint. *Teams Channels* can be created, for example, for each project (on the screenshot, it is called "CIA CPPS"). The users can invite their colleagues to the channel to communicate via persistent messages. Each of the channels is linked to a dedicated SharePoint Site's folder structure under the tab "Files" (c.f. Figure 2.7).

Additionally, under the tab "Tasks", a user can create a task board to manage to-dos. We created three buckets for demonstration purposes - *To Do*, *Review*, and *Done*. The users can label tasks with various labels, e.g., the "CR1" task has a label to document that it is a change request.

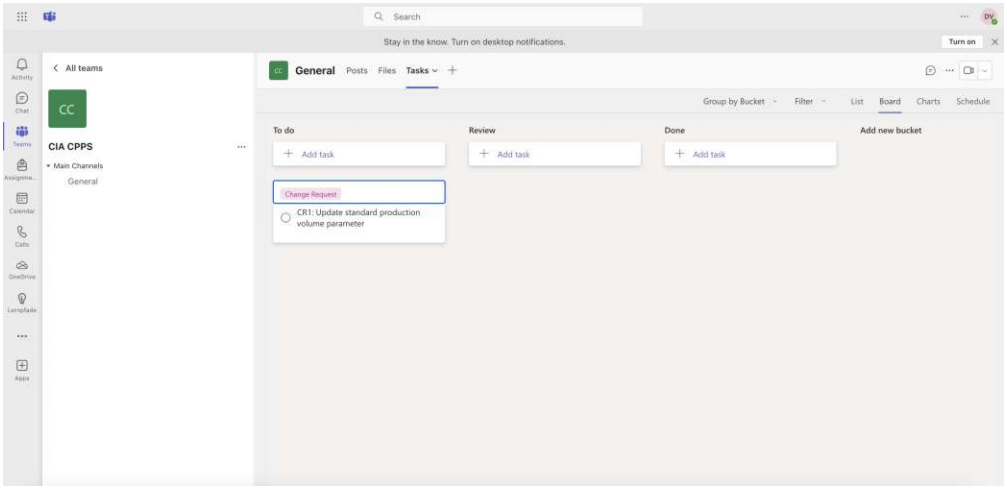

Figure 2.7: Tasks in *Microsoft Teams* for the *CIA CPPS* project evaluation.

2.5 Graph Theory Fundamentals

This section introduces the basics of graph theory necessary for the proposed system design. Graph theory is a study of *graphs*, which are structures of objects and relations between them. A graph consists of two elements: *vertices* representing objects and *edges* representing relations. Graph theory distinguishes between *directed graphs*, represented by arrow edges, and *undirected graphs*, represented by plain line edges.

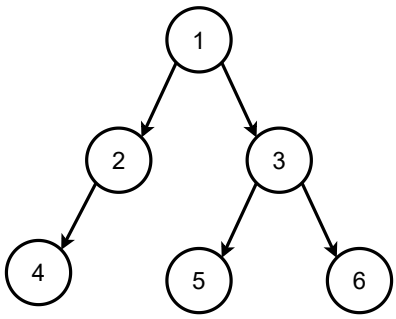

Figure 2.8: An exemplary directed graph.

Figure 2.8 shows an exemplary directed graph. The knowledge of a CPPS is represented as a directed graph in the proposed M-CIA solution approach. Such mathematical structures can be traversed using *depth-first* or *breadth-first* search algorithms.

Depth-first search explores the graph as deep as possible along each branch before backtracking to the upper levels. An exemplary depth-first search graph traversal on the graph in Figure 2.8 would be visiting the root node one at the depth level 0 and exploring it. Then, its direct neighbor vertex *2* would be explored (depth level 1), and vertex *4* would be explored (depth level 2). Here, the algorithm backtracks to vertex *2*, as the vertex *4* has no more adjacent vertices. Again, vertex *2* has no more vertices to explore. Therefore, it backtracks. The algorithm repeats this procedure for vertex *1* and sees that vertex *3* was not explored yet. It explores vertex *3* and its adjacent vertex *6*. The algorithm continues the procedure until there is no unexplored vertex.

Breadth-first search explores the neighbors of the vertices at the current depth before moving to the next depth level. An exemplary breadth-first search graph traversal on the graph in Figure 2.8 would first visit the root node *1* at the depth level 0, then explore all of its neighbors. The exploration can be performed in any order. The algorithm could visit first vertex *2* and then vertex *3* (depth level 1). Then, it can select vertex *2* for exploration. The algorithm visits neighbor vertex *4* (depth level 3). Then, it selects the vertex *3* for exploration. Finally, it visits its neighbors *5* and *6* (depth level 3). Such traversal algorithms are important when working with data represented by graphs.

The knowledge graph created as part of the solution approach to represent the CPPS knowledge is traversed in a *dept-first-search* fashion to identify change impact.

Graphs are the foundation of knowledge graphs used for knowledge reasoning. To implement knowledge graphs, a typical relational database system is not the best fit to represent graph data and conduct graph exploration. There are various graph databases for different purposes, which can be categorized into *Resource Description Framework (RDF) Triple Stores*, and *Labeled Property Graphs*. Both have the same purpose of storing and providing a query mechanism for interaction with the data but implement it differently.

RDF Triple Stores represent the data as triples in a subject-predicate-object structure, where the subject and object are graph nodes, and the predicate is an edge. In Labeled *Property Graphs*, the entities and relations have attributes represented as key-value pairs. Example of a RDF graph database is GraphDB[4], and an example for Labeled Property Graph is Neo4j[5], or ArangoDB[6]. RDF database systems all support the standard SparQL[7], while Labeled Property Graphs have their query languages. Neo4j queries are written in Cypher Query Language[8] and ArangoDB has ArangoDB Query Language (AQL)[9].

[4]GraphDB: `https://www.ontotext.com/products/graphdb/`
[5]Neo4j: `https://neo4j.com/`
[6]ArangoDB: `https://www.arangodb.com/`
[7]SPARQL: `https://graphdb.ontotext.com/documentation/10.0/devhub/sparql.html#using-sparql-in-graphdb`
[8]Cypher Query Language: `https://neo4j.com/developer/cypher/`
[9]ArangoDB Query Language: `https://www.arangodb.com/docs/stable/aql/`

To build a multi-domain change dependency graph to enable the iterative exploration of the change impact in a CPPS, the proposed system design will include a labeled property graph database.

CHAPTER 3

Related Work

This chapter summarizes the basics of CPPS engineering, the state-of-the-art knowledge representation in CPPS, change management in software and CPPS engineering and related work in Git- and DevOps in CPPS engineering.

3.1 Cyber-physical Production Systems Engineering

To explain the concept of Cyber-Physical Production Systems, we first introduce the CPSs. CPSs are systems of collaborating computational entities intensively connected with their physical surroundings and processes, providing and utilizing data available on the internet [Monostori, 2014]. Such systems include autonomous cars, robotic surgery, intelligent buildings, smart electric grids, and smart manufacturing [Monostori, 2014]. The latter example bridges the definition of CPPS, as CPPS are cyber-physical systems used in production and manufacturing. According to the definition of Monostori [2014], CPPS *"consist of autonomous and cooperative elements and sub-systems that are getting into connection with each other in situation-dependent ways, on and across all levels of production, from processes through machines up to production and logistics networks."*

The expectations of CPPS are expectations typical for smart systems, such as self-organization, self-maintenance & self-repair, remote diagnosis, real-time control, autonomous navigation, transparency, predictability, and efficiency [Monostori, 2014]. Resilience and reconfigurability are significant characteristics of CPPS, provided by the *self-x production* capabilities [Prenzel and Steinhorst, 2021].

Due to various reasons, organizations must transform their CPPS to stay competitive. Koch et al. [2016] define several sources of change, such as factory lifecycle (e.g., aging of manufacturing resources), manufacturing change, which can also trigger subsequent engineering changes, factory-internal causes (e.g., non-fulfillment of manufacturing requirements, mistakes in production planning), product lifecycle (e.g., a varying number

of units, change of product mix, the introduction of new product variants), engineering change, product-related causes (e.g., quality or design issues), company internal and external causes (e.g. laws and regulations, stricter environmental or labor regulations, new norms, standards), business operations (e.g. performance improvement, new KPIs), customer feedback, technology-related causes (technology evolution), and procurement (e.g. changing suppliers, different materials).

Koch et al. [2016] also specify two types of changes: manufacturing and engineering changes. Manufacturing changes describe any change, such as adaptation or reconfiguration within the production system, and comprise relevant attributes such as cost and duration [Koch et al., 2016]. On the contrary, an engineering change is any alteration made to the system assets and its artifacts, such as drawings, or software that has already been released to manufacture a product [Koch et al., 2016]. Such change comprises the addition, removal, or substitution of existing system parts and changes to the system or asset parameters. This thesis focuses on the latter, the engineering changes in the asset parameters.

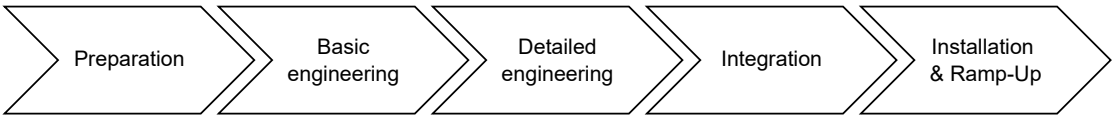

Figure 3.1: CPPS engineering process, based on Eckhart et al. [2019].

Figure 3.1 shows the basic CPPS engineering process that consists of five phases, which may overlap in time and require a close collaboration of stakeholders from multiple disciplines, based on Eckhart et al. [2019].

In the preparation phase, objectives, requirements, and system processes of the CPPS are planned. The preliminary CPPS design is created in the basic engineering phase, and the system components are selected. Then, the stakeholders from various domains, such as mechanical and electrical domains, perform the detailed planning of the production system.

Next, the planned components are purchased, and the system is constructed, configured, and tested. Finally, the production system is installed on-site and commissioned. Francalanza et al. [2017a] adds production, CPPS maintenance, and factory disposal/refurbishment to the design and engineering phases depicted in the figure to describe the whole CPPS lifecycle. This related work provides the fundamental knowledge of the CPPS lifecycle that informs the thesis to frame the application potential of the proposed M-CIA method throughout the lifecycle phases.

3.2 Knowledge Representation in CPPS

As mentioned in Chapter 2, knowledge modeling and representation are a crucial aspect of the thesis. Depending on the use case, there are several approaches to creating knowledge representation of CPPS.

Firstly, Systems Modeling Language (SysML) is a modeling language for systems engineering, an extension of Unified Modeling Language (UML) from software engineering. On the other hand, AutomationML (AML) has been created as an extension of XML and an open standard to enable (plant) data exchange in heterogeneous cross-domain systems [Meixner et al., 2021a].

However, these approaches strongly focus on automation code and machine-to-machine exchange. It is possible to model system parts, but products and processes are not part of out-of-the-box approaches. Modeling capabilities that include the domain-specific aspects are also missing.

[VDI/VDE 3682] is a guideline that suggests modeling CPPS as a set of three types of assets: products, processes, and resources. Domain experts and basic planners in CPPS engineering design these assets that form a CPPS, their functional relation, to specify valid production process and resource designs that fulfill the customer requirements [Meixner et al., 2021b]. This thesis will use the PPR modeling approach to create the knowledge representation of the CPPS.

The CPPS assets have various dependencies, and it is difficult to represent these dependencies sufficiently in heterogeneous artifacts, such as system plans, models, and tool data, to coordinate the implementation of changes to shared asset properties in a multi-disciplinary environment [Biffl et al., 2021]. To represent production knowledge with changes, Biffl et al. [2021] introduced the Product-Process-Resource Asset Network (PAN) coordination artifact, a knowledge graph based on I4.0 assets.

Similarly, [Rinker et al., 2021] proposed a Multi-Domain Engineering Graph (MDEG) which depicts the PPR assets, their properties, and their domain affiliation. To complement such visual models and make them processable by machines and provide formal definitions, Meixner et al. [2021b] presented a design of a PPR DSL, which will be used in this thesis. The initial PPR DSL was extended for the notion of concepts and common concepts [Grangel-González et al., 2020, Rinker et al., 2019] and is available publically as MDM-CPPS IDE in *VS Code Marketplace*[1]. This work will extend its underlying MDM-CPPS DSL for (common) concept parsing capabilities and writing capabilities of all language elements to enable the processing of the models. The concept idea is that each domain defines the concept in their *Concept Glossary* and later negotiates the common concepts as a consensus of involved domains. The resulting glossary is called *Common Concept Glossary*.

The processes of the CPPS organizations can be modeled using Business Process Model and Notation (BPMN), as it offers a standardized graphical notation based on a flowcharting technique that allows stakeholders to easily specify the behavioral view of their system in terms of business processes Falcone et al. [2017]. BPMN provides an intuitive notation for business analysts and designers, who specify the business process, and technical users who eventually implement complex systems based on the specified process

[1]MDM-CPPS IDE: `https://marketplace.visualstudio.com/items?itemName=ModelIEE.mdmcpps-ide`

Falcone et al. [2017]. This thesis uses BPMN to model the initial process diagram for the evaluation use case in the fertilizer production industry.

Another approach to knowledge management in CPPS is Semantic Web Technologies (SWT) and ontologies. Biffl and Sabou [2016] has conducted case studies in different engineering domains to demonstrate the use of SWT in intelligent engineering applications and also provide a guide on how to apply SWT in multi-disciplinary engineering settings. Additionally, Hildebrandt et al. [2020] presents an ontology-building method that is adjusted to the needs of the CPS in the manufacturing domain. The authors also present a reusable set of ontology design patterns that have been developed with the ontology-building method on an industrial use case, which is another implementation of the [VDI/VDE 3682], conceptually similar to the PPR and CC taxonomy.

3.3 Change Management in Software Engineering

One of the early definitions of CIA dates to 1996 and is described in detail in software engineering by Bohner and Arnold [1996]. The authors define the CIA as *"identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change"*. The thesis utilizes this definition to design a multi-domain CIA method in the multi-domain CPPS setting.

Changes to software code are inevitable as software systems grow in size and complexity, making CIA a critical tool in controlling changes [El Nemr and Elzanfaly, 2018]. The impact analysis process should have the tracing capability to trace a change from the requirement models down to specific source code elements, and vice versa [El Nemr and Elzanfaly, 2018]. Additionally, the authors say that a change is not limited to one software artifact. Still, it often impacts various system life-cycle objects, such as architectural models, requirement models, and source code, which makes a CIA challenging. It is important to note that dedicated CIA approaches exist for each of those model types. Therefore, the authors propose a framework for CIA in software engineering that combines multiple scopes of system development artifacts models and utilizes traceability between the artifacts to extract hidden links and ripple effects [El Nemr and Elzanfaly, 2018].

Their framework is based on the Model-View-Controler architecture and utilizes a graph database to represent the objects and their dependencies. The approach is rather theoretical but points out an interesting aspect. If various stakeholders use the framework in any development phase, the graph query results must be adjusted to the stakeholder group. El Nemr and Elzanfaly [2018] give an example that a software engineer might be interested in concrete lines of code impacted by the change. At the same time, a team lead may inquire about an overall view of the affected source code to estimate the required implementation time. The approach proposed in the thesis will also facilitate traceability between various domain-specific models to extract links and identify the effects. Also, the solution approach proposed in the thesis will use a graph database to represent objects (CPPS assets) and their dependencies.

In software engineering, the system models are often described using UML. To interact with and explore the models, the *UMLtoGraphDB* was proposed Daniel et al. [2016]. The authors argue that several solutions already exist to transform the UML models to SQL databases. However, graph databases provide advanced and expressive query languages optimized to traverse highly interconnected data. To facilitate the generation of a knowledge graph in a graph database from an MDEG based on its corresponding PPR DSL files, a similar approach has to be designed and implemented.

Wan et al. [2016] also propose a multi-perspective CIA method, by utilizing the SWT to address dependencies between heterogeneous software artifacts. The authors use the semantic web to construct ontology-based software engineering-linked data, which links the software engineering artifacts, such as requirements, code, bug reports, Git commits, and test cases [Wan et al., 2016]. Then, they build a weighted change impact matrix/graph using the dependency information extracted from linked data. Finally, they apply a change impact propagation algorithm and analyze the change impact [Wan et al., 2016]. The approach proposed in the thesis will use the concept and common concept approach and PPR to retain the semantic meaning and relationships between the assets.

Software engineering practitioners manage changes in software artifacts, especially of the source code and software configuration, in Git-based repositories and online developer platforms, such as GitHub and GitLab [Cosentino et al., 2017].

The changes to the system are driven by *issues* (change requests) that contain the requirement descriptions to be implemented. The practitioners implement the changes using various Git branching strategies, such as *Feature-based Workflow*[2], in which each feature is developed in a dedicated Git branch, or *Gitflow Workflow*[3] which extends the feature branches for specific-purpose branches such as development, hotfix or release branch.

Regardless of the workflow, the changes are merged into the main branch for the release into production after a peer review conducted via a coordinated process using *Pull Requests* (GitHub term) or *Merge Requests* (GitLab term). This thesis will utilize Git technology to manage text-based changes to the PPR DSL files, which represent models of the CPPSs.

Li et al. [2013] have surveyed 23 code-based CIA techniques proposed between 1997 and 2010 and proposed a comparative framework to help select the appropriate technique. Similarly, Lehnert [2011b] has reviewed 150 CIA approaches and proposed a new taxonomy of the CIA techniques, which helps to structure the field of CIA in software engineering. The proposed taxonomy comprises five categories: source code, architecture, requirements, miscellaneous artifacts, and combined scopes. The majority of the CIA approaches focus on the code CIA.

[2]Feature-based Workflow: `https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow`

[3]Gitflow Workflow: `https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow`

CIA is a well-established method in software engineering. Therefore, we will base our research on past knowledge and techniques in this field to design and evaluate our method for multi-domain CIA in a CPPS.

3.4 Change Management in CPPS Engineering

Francalanza et al. [2017b] address the challenge of factory changeability, as factories have long lifecycles that must evolve as the products they manufacture evolve. Over time, the product families evolve in the form of the addition or subtraction of system parts or parts' features. The manufacturing system designer has to consider not only the current product requirements but also the consequences of the design decision on the system's future capability and changeability.

Therefore, Francalanza et al. [2017b] identify two types of factory lifecycle consequences: Manufacturing Capability Consequences (MCC), the capability of the system to produce according to the requirements, and Factory Changeability Consequences (FCC), the capability of a system to change or modify to meet future system requirements. To explicitly reveal decision consequences on the Capabilities by providing the changeability knowledge, they propose the Changeability Knowledge-Based Product Development Approach Framework and implement it as a computational tool. The tool assists the factory and product designers in becoming aware of unintended MCC and FCC, influencing the capability and changeability of the factory. The solution was evaluated with a case study on a fictitious company with 25 stakeholders with experience in product development, and the preliminary results were positive. However, this approach does not directly address the multi-domain aspect of the CPPS engineering, and the target stakeholders are production designers in the factory design phase.

Bauer et al. [2017] propose a model-based CIA method in factory systems. They define the CIA as *"determination of impacted elements resulting from a proposed factory change and the prediction of the impact of all changes (initial and consequent) on selected manufacturing metrics"* such as throughput, or overall equipment effectiveness. The goal of the proposed method is to enable the stakeholders to analyze a quantified change impact on the manufacturing metrics. Bauer et al. [2017] states that the current methods only analyze the impact of the initial change set. Still, if consequent changes have to be performed, their impact is analyzed by repetition of the approach, and the planning of consequent changes is not integrated into the procedure.

Figure 3.2 depicts their procedure for identifying impacted elements and consequent changes. First, the changes are made to the system model, then identified and classified based on change types. Second, the relevant relations that define the possibly impacted direct neighbors are identified, and the constraints are verified. Finally, other change impacts that the definition and verification of constraints cannot predict are reviewed. The thesis leans on this Procedure to integrate the planning of consequent changes and their consequent impact.

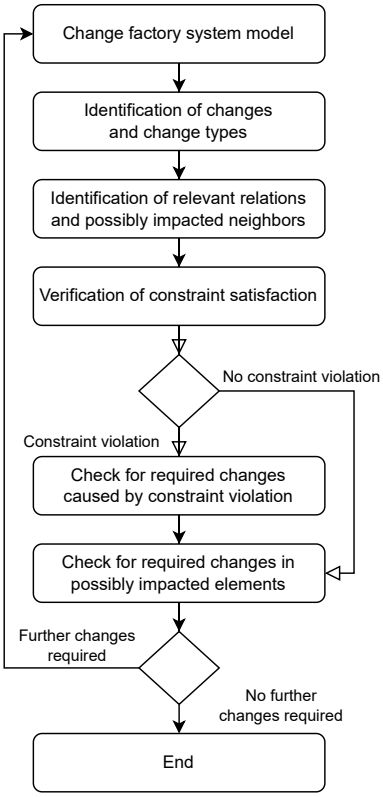

Figure 3.2: Procedure for identifying impacted elements and consequent changes, as proposed by Bauer et al. [2017].

Hoang et al. [2018] address the impact analysis by first modeling the system based on the VDI/VDE 3682 guideline, which requires that each process has at least one input product, information, or energy that is transformed during the process into at least one output element. A technical resource performs the process. Then, create a Multiple-Domain Matrix (MDM) that represents the inter-dependencies between the process, its technical resource, and the input and output elements on the parameter level.

Finally, a rooted graph tree is constructed to analyze the adaptation options and their impact on other system's elements. The rooted graph tree depicts the dependencies of the Multiple-Domain Matrix for better analytical capabilities. The approach was evaluated with a case study on a scenario from cigarette manufacturing to analyze if the current machine setup could work with the newly requested material of an input product. This work does not address the multi-domain aspect of CPPS engineering. In addition to addressing this aspect, this thesis will adopt the concept of modeling the system based on the [VDI/VDE 3682] and define the inter-dependencies of the system on the parameter level, as specified in previous sections.

Heinrich et al. [2018] proposed a generic methodology for *domain-spanning* CIA and also rely on a model-based approach to estimate the impact of a change before the real system

is modified. Their methodology consists of domain-independent and domain-specific elements. The domain-independent elements include [Heinrich et al., 2018]:

Domain-independent metamodel of modification provides a metamodel for describing the initial change (seed modification) and elements potentially impacted by the seed modification. Furthermore, the metamodel provides means to describe the change propagation from the *causing element* to the *affected element* and is referred to as *propagation step*. Finally, the task list is defined, and the task element includes information about the seed modification and a set of propagation steps.

Task List Algorithms is used to derive and manage the task list. The authors define several types of algorithms, such as the Algorithm for Derivation of Task Lists, which is used for analyzing the difference between the base and target (changed) model; the Algorithm for Duplicate Elimination, which is used to check duplicate tasks and eliminate them; and the Algorithm for the Task List Sorter, which improves the comparability between two lists by sorting them.

Decision Support is used when it is beneficial to include the knowledge of a domain expert not contained in the models to reason about the task list. The authors define the *Metamodel of Decision Support*, which facilitates the marking of the tasks with human decisions such as *Confirm, Exclude,* and *No Decision* (default). Additionally, the authors define *Algorithm of Decision Support* for excluded tasks to prevent them from triggering derivation of consequent tasks in future iterations.

These domain-independent elements are reflected in domain-specific elements. The most relevant domain-specific elements from the proposed method are described below, as proposed by Heinrich et al. [2018]:

Metamodel of System describes the systems in a given domain based on state-of-the-art system modeling methods and tools.

Domain-specific Metamodel of Modification is an extension of its domain-independent counterpart and involves specifying the seed modifications, potentially affected elements by changes, and dependencies.

Algorithm of Change Propagation Analysis specifies the rules for change propagation. It is specified for the elements of the metamodel of the system and the domain-specific metamodel of modification.

This thesis applies the notion of domain-independent and domain-specific elements and their definitions to facilitate the application of *Git Workflow* for multi-domain CIA.

Meißner et al. [2021] use model-based systems engineering for rapid engineering change management on the parameter level. Their approach incorporates the links between

dependent system parameters and the domain-specific models, which allows quick estimation of change request impact. Additionally, they show that automation of the execution of engineering changes is possible if the domain-specific models are fully parametrized [Meißner et al., 2021]. The authors create the system model using SysML. The proposed methodology addresses the following issues: 1) recognizing the need for changes as early as possible, which they do by modeling requirements in SysML where the requirements are verifiable at any moment, by connecting the system parameters using the *satisfy* connector between the parameter and a requirement in *requirement diagram*, and 2) through linkage of the domain-specific models with the main SysML model, the parameter changes can immediately be propagated into the corresponding external models. The authors state that, e.g., MATLAB[4] or ModelCenter[5] has to be used as an interface to execute external (domain-specific) models, as the majority of the external models cannot be integrated directly. This thesis applies the concept of linking the domain-specific models to the common model to enable propagation.

3.5 DevOps and GitOps

DevOps is a well-established approach in software engineering to minimize lead time and maximize the quality of the software. There is also an interplay between Agile and DevOps, as DevOps can be seen as an enabler for Agile development. Agile development aims to deliver valuable product increments in short iterations. DevOps aims to break silos between Development and IT Operations teams, which requires a "shift" in mindset compared to traditional software engineering. Firstly, it is "shift-left", which means developing and testing the product increments in a production-like environment. Secondly, "shift-right" requires small product increments to be deployed to production as soon as possible, reducing the risks and lowering the testing effort.

Recently, the applicability of DevOps to CPPS engineering gained the interest of the research community [Ugarte Querejeta et al., 2020, Koren et al., 2023, Hegedüs et al., 2021]. As motivated in Chapter 1, production and manufacturing enterprises face changing markets and must transform their assets to stay competitive. The market requires flexible, reconfigurable, and customized production systems capable of adapting to changes throughout the product lifecycle [Ugarte Querejeta et al., 2020]. To fulfill this requirement, new approaches are necessary that decrease development costs and time [Ugarte Querejeta et al., 2020]. DevOps helps software engineering enterprises to achieve this goal.

However, there are several challenges to benefiting from DevOps in CPPS engineering. Therefore, DevOps, as we know it from software engineering, has to be adapted to the CPPS field. has mapped the challenges cite10128073 and Kreutz et al. [2021]. Exemplary challenges identified by these authors are multi-disciplinarity, agility, and the interconnection of hardware and software. There are silos between engineering

[4]MATLAB: `https://www.mathworks.com/products/matlab.html`
[5]ModelCenter: `https://www.ansys.com/products/connect/ansys-modelcenter`

domains; integration of information can be hard due to the variety of technical languages and frameworks used [Koren et al., 2023]. Agility is hard to achieve due to safety and security regulations [Koren et al., 2023]; certifications are issued for whole systems, and the certification process is lengthy, which contradicts the idea of short release cycles. Hardware and software have different lifecycles, and achieving backward compatibility between them adds complexity to the development process [Kreutz et al., 2021]. However, the authors also provide initial solution proposals to overcome identified challenges as the potential of DevOps in CPPS engineering is promising.

Adoption of DevOps in CPPS engineering would not only require breaking silos between Development and Operations but also unifying many other engineering disciplines, such as mechatronics and electronics. There is a need for all engineers to gain awareness of how changes in one discipline affect the other disciplines related to the product [Kreutz et al., 2021]. This thesis builds on this gap and aims to foster this awareness of CPPS engineers with its results.

The above-mentioned related work provides building blocks for the thesis, and the thesis will apply some of the DevOps practices, such as Continuous Integration, Continuous Testing, and Infrastructure as Code. Additionally, Multi-view Change Management Workflow from Rinker et al. [2022], introduced in Chapter 2, on which our research is based, revolves around Continuous Integration.

Another widely use practice from software engineering is GitOps. In this practice, the engineering teams have one single source of truth of the system and its configuration, which is managed with a version control system Git [Beetz and Harrer, 2022]. This implies that the system environments are operated strictly based on the contents of the files stored in Git, including creating, changing, and destroying the system environments [Beetz and Harrer, 2022]. This thesis partially utilizes GitOps practices, as we will use Git-based repositories and define the system models using the domain-specific language files, which will be the single source of truth.

CHAPTER 4

Methodology

This chapter presents the methodology of the thesis project. Section 4.1 introduces the research questions, Section 4.2 presents the *Design science* methodology, and Section 4.3 instantiates the M-CIA Framework's activities to outline the concrete plan along the PIAs and PDAs.

4.1 Research Questions

This section presents and motivates the preliminary research questions. Each question includes the outline of how the question will be addressed and what results to expect. The questions also briefly describe how the results will be evaluated. Figure 4.1 visualizes this information and represents the research methodology activities, questions, and the expected results.

In the literature, there are recurring mentions of the industry facing challenging markets and changing customer requirements, pushing manufacturers into engineering changes and transforming their systems to be reconfigurable and resilient [Francalanza et al., 2017a].

Complex engineering changes and multi-domain coordination are inevitable to re-engineer production systems. To succeed in such a complex effort, change management and engineering CIA must be in place.

These statements are supposed to motivate the change management research in CPPS, and we find various approaches and methods to CIA. However, challenges still seem to exist, especially in multi-domain systems engineering with heterogeneous artifacts [Mengist et al., 2021]. Therefore, we aim to understand the current multi-domain CIA practices and have to answer the following research question:

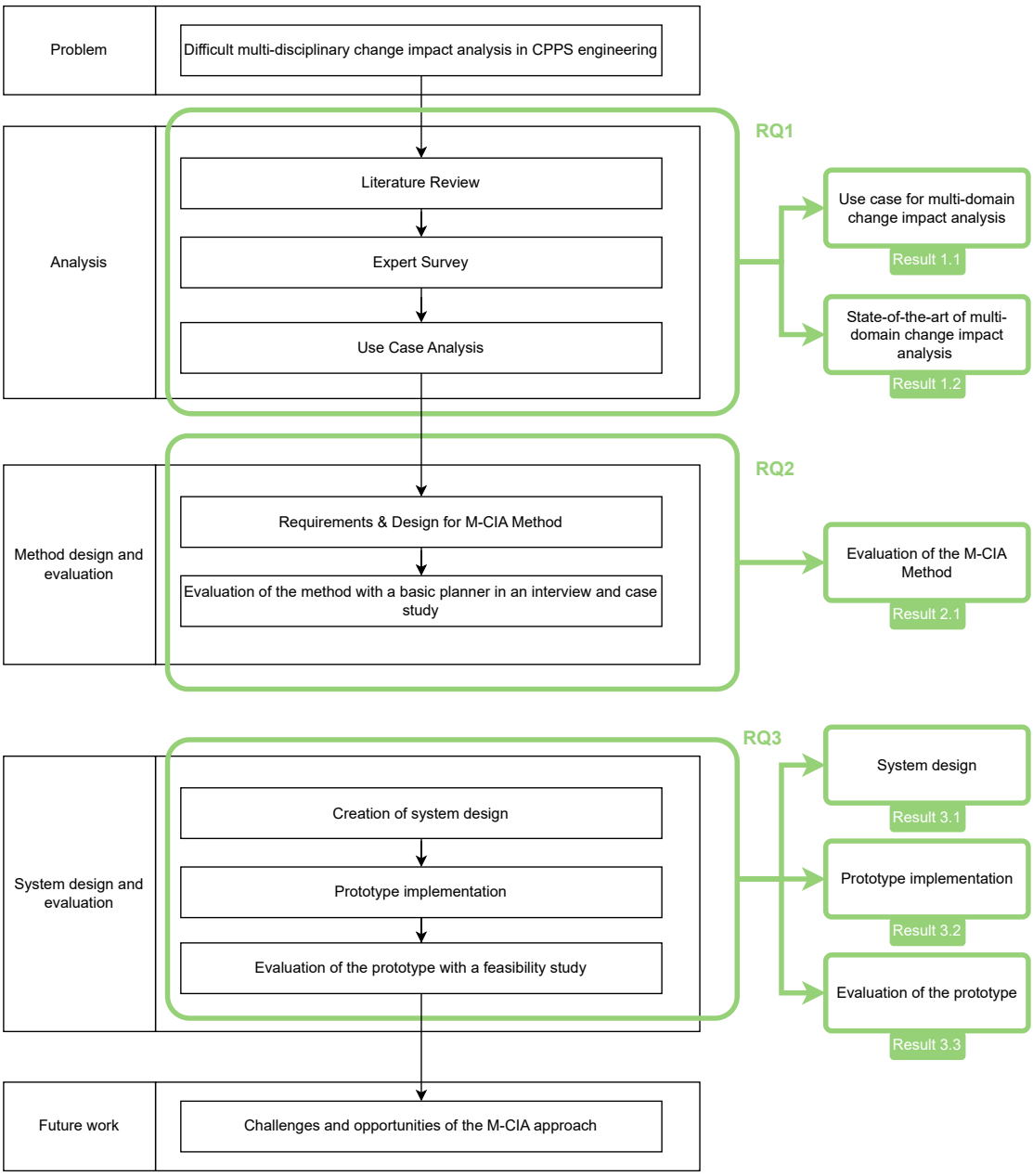

Figure 4.1: Research methodology activities.

Research Question 1: What is the current state of CIA in multi-domain cyber-physical production systems engineering? This research question aims to understand how industry experts and researchers perceive change management and CIA in a multi-disciplinary environment in the context of engineering their CPPS.

To answer the research question, we aim to survey industry experts and researchers,

focusing on their experience with change management and their approach to CIA, to provide insights into current practices. Additionally, we will conduct a state-of-the-art literature survey informing research on existing methods and approaches to the CIA in the context of CPPS engineering.

We also want to elicit the perceived applicability of the *agile Multi-view Change Management Workflow* [Rinker et al., 2022] from the state-of-the-art survey, on which the proposed solution approach will be based on in this research project. The quality of the expert survey will be ensured by *friendly* experts, who will answer the initial versions of the survey to ensure understandability and reproducibility.

Finally, the research question results will present the survey results, depict them using a meta-analysis, and draw conclusions that inform and motivate the thesis.

As described in Chapter 2, a lot of research has been done in the field of CIA in software engineering. Software engineering teams adopt agile collaboration methods that enable (parallel) work on small increments of the system and regularly integrate them. Therefore, a CIA is vital to the quality of the increment. As described in Chapter 5, parallel work on the same engineering data artifacts in CPPS engineering is inevitable. Therefore, we aim to understand how we can adopt CIA practices from agile software practices engineering in our use case.

Hence, we want to answer the following research question:

Research Question 2: What methods from agile Software Engineering facilitate applying the Git Workflow for efficient multi-domain CIAs in CPPS engineering? This research question aims to understand the current state of CIA practices and methods that work well in the software engineering discipline and use them to apply the Git Workflow in production systems engineering to enhance multi-domain CIA.

As described in Chapter 3, Lehnert [2011a] has conducted an extensive structured review of 150 software CIA approaches. Recent work of Mengist et al. [2021] also addresses this issue in systems engineering by applying software engineering methods and tools, such as Git for version control, RESTful Services for data exchange, and graph database. Therefore, it seems promising to consult the state-of-the-art of CIA in software engineering to address the issue in production systems engineering.

To address the research question, we will conduct a literature review of CIA methods from software engineering and apply the most promising ones to the use case defined in Chapter 5 and extend the previous work of Rinker et al. [2022]. The extension of the previous work will address 1) the specification of who is involved in the CIA and how to establish this knowledge, 2) the definition of how the CIA review process is coordinated, and 3) how the stakeholders are notified about the necessary actions and results.

Finally, we will design the method on the illustrative use case and explore the change dependencies of the CPPS in discrete manufacturing to show the method's feasibility.

The method will also be validated on an evaluation use case from batch manufacturing to observe the suitability of the approach beyond discrete manufacturing and to evaluate the method's efficiency. We define the efficiency in terms of the execution time of the most relevant tasks in the CIA process and the perceived improvement of the method to the relevant stakeholders.

Based on the method from addressing research question 2, we will propose a system design for efficient and automated CIA in a multi-domain setting. This motivates the last research question:

Research Question 3: What system design and architecture can efficiently facilitate conducting the Git Workflow-based multi-domain CIA method? To allow for the evaluation of a multi-domain CIA inspired by methods in software engineering, this research question aims to define an appropriate information system design and architecture that will support the method execution and evaluation.

We will address this research question by reviewing the architectures in the related work as part of the review in research question 2. Also, we will consider and incorporate state-of-the-art practices in software engineering regarding delivering information system architectures, such as service-oriented architecture, graph-based databases, and modern programming frameworks.

The feasibility of the system design and architecture will be evaluated by implementing a system prototype and executing the method on the illustrative use case.

4.2 Design Science Methodology

The *Design science* paradigm is a problem-solving paradigm with roots in engineering and the science of the artificial that seeks to *"extend the boundaries of human and organizational capabilities by creating new and innovative artifacts"* [Hevner et al., 2004]. The paradigm characterizes the research in the Information System (IS) discipline to a great extent [Hevner et al., 2004].

Design science defines three closely related cycles of research activities [Hevner, 2007], as shown in Figure 4.2: *relevance*, *design*, and *rigor cycle*. According to Hevner, the relevance cycle connects the environment and the context of the IS research project to the IS research activities and motivates the research activities with business needs. The rigor cycle connects the IS research activities to existing research expertise, theories, and methodologies, providing applicable and past knowledge. The design cycle is the core of the IS research, iterating between the development and evaluation activities in the research project.

Relevance cycle. We will collect requirements from domain experts in the multi-disciplinary environment in CPPS engineering that motivate the need for multi-domain CIA, which gives this thesis relevance. We collaborate with our industry partners and

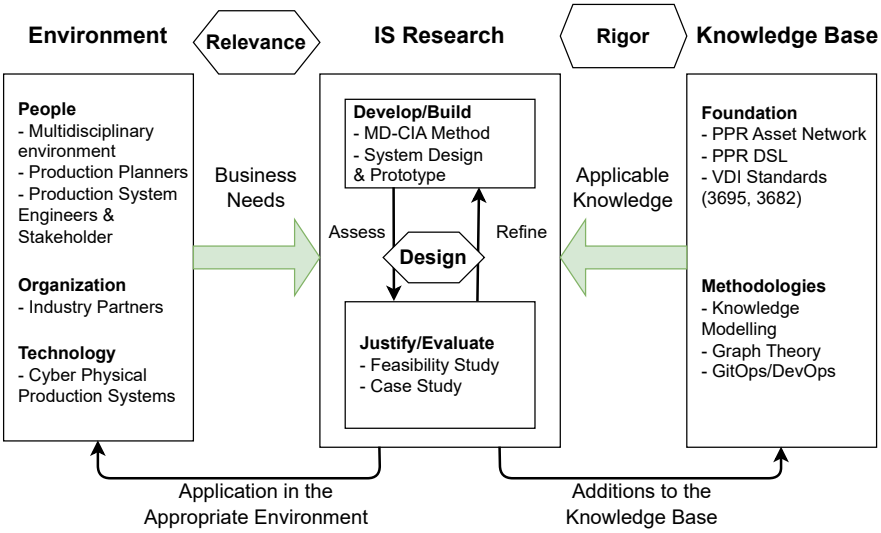

Figure 4.2: Information Systems Design Science Research Framework [Hevner, 2007] adapted to the thesis.

research center throughout the research project to define relevant use cases based on which we design and evaluate the solution approach. The use cases will focus on CPPS, which are systems consisting of products, processes, and resources and perform discrete and batch production.

Rigor cycle. *Design science* draws from existing scientific theories and methods [Hevner, 2007]. Therefore, well-established approaches to CIA in software engineering and CPPS engineering will provide us with the scientific foundation and relevant knowledge.

Specifically, the foundation will be the PAN[Biffl et al., 2021], PPRDSL[Meixner et al., 2021b], and relevant VDI Standards. System knowledge must be modeled to describe the CPPS. Therefore, knowledge modeling methodologies will be applied. Additionally, foundational concepts from graph theory will be used to transform the system knowledge into an interactive graph that can be traversed and queried. Finally, practices from software engineering that facilitate agile collaboration, such as Git- and DevOps, will be applied.

The addition to the knowledge base will be the state-of-the-art elicitation of the CIA practices in a multi-domain setup in the field of CPPS engineering. Further, we will design a method for multi-domain CIA for stakeholders from the field of CPPS engineering. The method will be evaluated on feasibility and efficiency. We will also contribute with an appropriate system design to execute the method and corresponding prototype.

Design cycle. According to Wieringa [2014], the design cycle is decomposed into three tasks, problem investigation, treatment design, and treatment validation, that are iterated over by the researches in design science research projects. Based on the business

needs and applicable knowledge elicited from the relevance and rigor cycles (problem investigation), we will iteratively design a method to address the problem defined in Chapter 1 (treatment design).

For the resulting method, we will propose a system design and a prototype that supports and automates the method (treatment design). Finally, we will evaluate the solution with a feasibility and a case study (treatment validation).

4.3 M-CIA Framework as a Guideline

As described in Chapter 2, the thesis follows the M-CIA Framework [Rinker et al., 2023a] to design and evaluate a novel solution approach. The M-CIA Framework was designed and published during this thesis project as a basis for further research on multi-domain CIA [Rinker et al., 2023a].

This section describes how the PIAs and the PDAs will be conducted in the context of the thesis, based on two leading questions. Additionally, this section describes what activities mainly address the research questions (c.f. Figure 4.3) and assigns each activity to the *Design science* cycles. The section is organized into two leading questions and depicts the M-CIA Framework with the research questions in Figure 4.3.

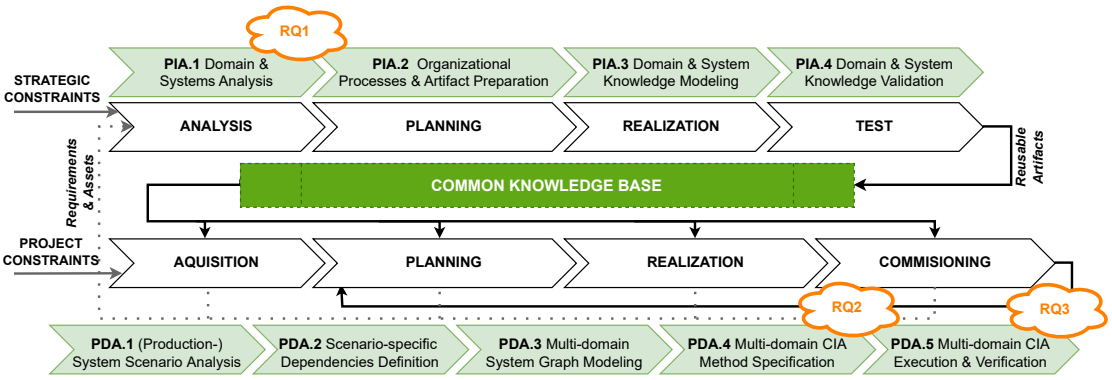

Figure 4.3: M-CIA Framework with research questions [Rinker et al., 2023a].

How will we apply the PIAs of the M-CIA Framework to enable CIA investigation project for a CPPS?

PIA.1 Domain & System Analysis. As defined in the research agenda of the previous publication Rinker et al. [2023a], this activity supports the acquisition of an overview of existing CIA approaches by conducting state-of-the-art analysis and an expert survey (RQ1). We have already conducted state-of-the-art analysis in Chapter 2 and Chapter 3 (RQ1). This activity belongs to the *relevance cycle*.

PIA.2 Organizational Process & Artifact Preparation. PIA facilitates the analysis of the organizational processes of the industry partner and establishes the understanding of their production processes. The raw data artifacts used by the industry partners, such as process diagrams and drawings of the production plant, will be gathered throughout this activity. This activity is part of the *design cycle* and its task *problem investigation*.

PIA.3 Domain and System Knowledge Modeling. This activity should result in the knowledge modeled as reusable artifacts. We intend to conduct first the modeling of the production process based on BPMN, and then transform the knowledge using the PPR modeling and MDEG to enable the *project setup phase* from Section 2.1. To conduct domain and system modeling and work out common concepts, various collaboration approaches seem promising, such as workshops or structured brainstorming. Alternatively, *Big Room Planning*[1] could be adjusted to our needs and goals 1) facilitating a cross-domain exchange, in which the artifacts are presented and each domain has to think of dependencies or missing information, 2) fostering awareness of dependencies in the organization, 3) achieving awareness of what artifacts will belong to the *Common Knowledge Base*. This activity is part of the *design cycle* and its task *treatment design*.

PIA.4 Domain and System Knowledge Validation. To validate the correctness of the reusable artifacts from the PIA.3, such as the identified CC and their dependencies, we will re-apply selected collaboration approaches from PIA.3 to assure the quality and completeness of the identified interdependencies between domains, assets, and processes. This activity is part of the *design cycle* and its task *treatment validation*.

Common Knowledge Base. The resulting reusable artifacts of the project-independent activities are collected in the *Common Knowledge Base*. The *Common Knowledge Base* will contain relevant documentation, models, and artifacts that we will later use to conduct the project-dependent CIA activities. The artifacts will be collected in a document-driven information system, such as *Confluence* and *MS SharePoint*. The models will be created using draw.io[2] and OmniGraffle[3]. Alternatively, model-driven tools, such as *Enterprise Architect* or engineering lifecycle management tools such as *IBM Engineering Lifecycle Management Suite* or *Siemens COMOS* could be used to collect models.

How will we apply the PDAs of the M-CIA Framework to conduct CIA investigation project for the CPPS lifecycle?

PDA.1 (Production-) System Scenario Analysis. As specified in the research agenda of the previous publication Rinker et al. [2023a], to enable a CIA investigation project, this activity facilitates the definition of the CIA use cases on which the further

[1] https://scaledagileframework.com/pi-planning/
[2] Draw.io: www.draw.io
[3] OmniGraffle: https://www.omnigroup.com/omnigraffle

project-dependent activities are conducted. The illustrative use case will be *Fasten Screw and Measure* (cf. Chapter 5) from the automotive industry. To evaluate the method and the system design, we will collaborate with a fertilizer producer to define the evaluation use case. A common view for the domains of the selected scenarios shall be created based on reusable assets from the *Common Knowledge Base*. In the context of the planned research, we will create an initial MDEG. We will create the initial MDEG as a knowledge representation of the common view for the domains identified in the selected scenarios based on the knowledge from the reusable assets in the *Common Knowledge Base*. This activity is part of the *relevance cycle*.

PDA.2 Scenario-specific Dependencies Definition. As part of our efforts in the PDA.2, we aim to integrate scenario-specific interdependencies of assets into the initially created holistic MDEG. To depict these dependencies, we will follow the concept of reactive links on the (common) concept- and property-level proposed by Raţiu et al. [2022]. This activity is part of the *design cycle* and its task *problem investigation*.

PDA.3 Multi-domain System Graph Modeling The PDA.3 facilitates the creation of an interactive machine-executable version of the knowledge representation in the form of a graph database, on which queries that represent relevant stakeholder concerns will be executed. We will use *Neo4j* for this purpose. This activity is part of the *design cycle* and its task *treatment design*.

PDA.4 Multi-domain CIA Method Specification PDA.4 facilitates method design for impact discovery based on the project requirements of the selected CIA use case. Based on Section 5, related work, and the results of the expert survey, we will define the requirements for such a method to answer the RQ2. This activity is part of the *design cycle* and its task *treatment design*.

PDA.5 Multi-domain CIA Execution & Verification. Finally, to verify the method we design in the PDA.4, we will validate the method and the prototype based on the system design on the selected evaluation use case from PDA.1 (RQ3). This activity is part of the *design cycle* and its task *treatment validation*.

CHAPTER 5

Illustrative Use Case

Section 5.1 depicts the context of the traditional multi-domain engineering process that motivates the research by employing the previous work of Biffl et al. [2019], Rinker [2021], as they have conducted an extensive analysis of the engineering processes in multi-domain systems engineering.

Further, Section 5.2 introduces the illustrative use case *Fasten Screw and Measure with a Robot Cell* that depicts a discrete production process from the automotive industry and provides the data basis to design the solution approach. This use case was initially introduced in the domain analysis of Meixner et al. [2021a].

Finally, Section 5.3 defines the minimal engineering and change management process, which will be the guiding process in the evaluation, conducted using the traditional artifact-based approach (c.f. Figure 1.1 and the newly proposed approach to provide a comparable set of evaluation tasks.

5.1 Context

Figure 5.1 shows simplified and abstracted engineering coordination of five stakeholders who work on CPPS engineering, each as part of their domain workgroup: basic planner, mechanical engineer, electrical engineer, automation engineer, and quality engineer. The engineers exchange *engineering data artifacts* representing the CPPS in artifact-based transactions. An example of an engineering data artifact could be CAD drawings, data sheets, various PDF files, spreadsheets, sketches, or AML and SysML models. Additionally, these artifacts are usually created in domain-specific tools and artifacts that support their work within their workgroup but make it hard to facilitate inter-disciplinary exchange [Meixner et al., 2021a].

Information exchange between different domains and departments is often document-based and coordinated via e-mail, leading to data inconsistencies [Meißner et al., 2021].

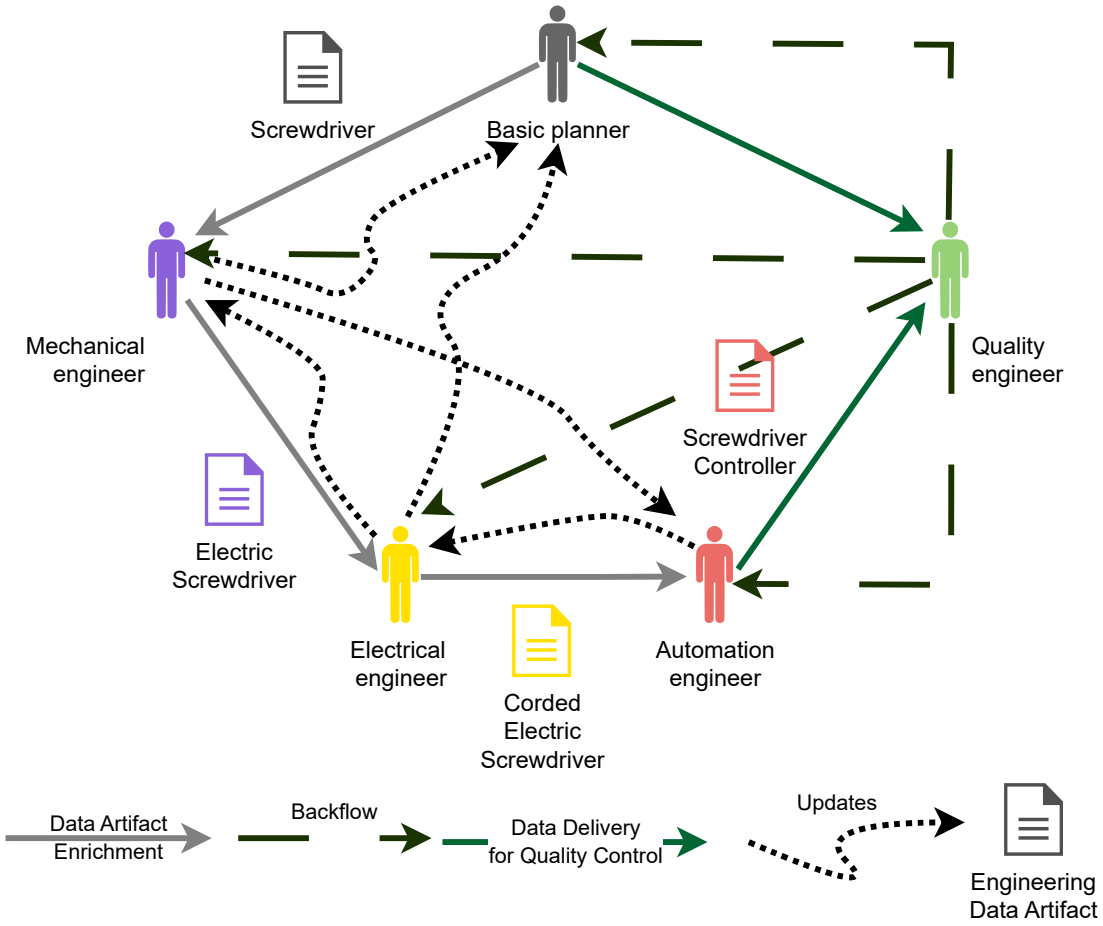

Figure 5.1: Illustrative cross-domain coordination of an engineering effort to engineer an electric screwdriver resource as described by Rinker [2021].

The figure is conceptually similar to Figure 1.1 introduced in Chapter 1. However, it focuses on exchanging artifacts related to a *Screwdriver* asset of a CPPS. An improved alternative to the traditional artifact-based approach to change coordination is previously described as agile Multi-view Change Management Workflow [Rinker et al., 2022] in Section 2.2.

In Figure 5.1, the basic planner conceptualized the initial functions and structure of the CPPS [Rinker, 2021]. In this scenario, the basic planner envisions the screwing capability to the CPPS. Therefore, a screwdriver will be part of the system. Secondly, the mechanical engineer constructs the system tree, incorporating mechanical functions, system components, properties, and the spatial arrangement of components based on the foundational plan [Rinker, 2021] (c.f. Data Artifact Enrichment in Figure 5.1).

Additionally, they specify that the screwdriver would be electrical. The electrical engineer follows the mechanical engineer's work and supplements the system by introducing

electrical components, establishing interfaces with mechanical elements, and specifying electrical system parameters, such as voltage or energy supply [Rinker, 2021].

These newly added and engineered system parts are later automated by an automation engineer who adds control code. In this case, the automation engineer would deliver a screwdriver controller that controls the screwdriver according to the production requirements. Finally, the quality engineer receives information from an automation engineer and a basic planner regarding the plant operation to assess the quality of the produced goods and the plant itself. This stakeholder provides feedback to the automation engineer and basic planner for further adjustments, which triggers the consequent updates. In a waterfall model of traditional project management, such a sequential engineering process would require minimizing the backflow of information and updates to artifacts from other disciplines (c.f. Backflow and Updates in Figure 5.1).

However, in reality, the engineering activities are carried out in parallel, which triggers necessary updates across domains due to dependencies between hardware- and software, regularly [Rinker, 2021]. To succeed in such a cross-domain engineering effort, an agile collaboration approach must help the engineering teams maintain a holistic view of the system and synchronize engineering artifacts during the engineering phase [Rinker, 2021].

In practice, the artifacts are integrated manually or with limited tool support, which can introduce risk and is prone to errors [Rinker, 2021]. Additionally, once there is a change in one artifact, figuring out what other artifact is impacted and should be updated accordingly is a cumbersome process.

5.2 Illustrative Use Case Fasten Screw and Measure

This section describes an exemplary industrial use case based on Biffl et al. [2021] using the MDEG notation [Rinker et al., 2023b].

The illustrative use case in Figure 5.2 depicts an important process of each automotive manufacturer and shows one robot work cell with one process, two products, and seven resources for a simple illustration of the problem. However, in a real car assembly plant, up to 300 robot work cells of up to 80 robot work cell types are derived from up to 30 different robot types [Meixner et al., 2021a].

Biffl et al. [2021] report that a compact robot work cell, that consists of 10 resources, is defined by up to 57 assets, asset properties, and engineering artifacts. Additionally, there could be up to 42 change dependencies between all of these objects.

In contrast, an extensive robot work cell comprising 38 resources is defined by up to 215 assets, asset properties, and engineering artifacts. These objects could depend on each other in up to 379 cases [Biffl et al., 2021].

The authors also report that in such a car assembly plant, engineers from up to 15 domains collaborate to define technical aspects of the production system parts and to

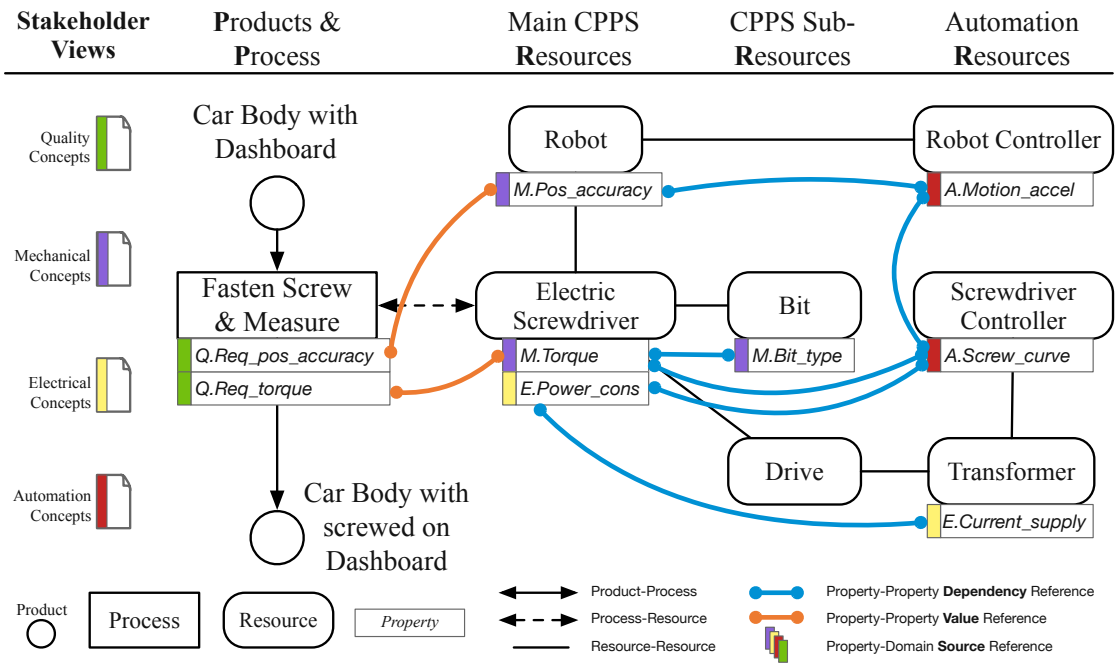

Figure 5.2: Illustrative use case *Fasten Screw and Measure* depicting a robot work cell in automotive manufacturing based on [Rinker et al., 2023b] in MDEG notation.

automate production processes. The current illustrative use case defines eight change dependencies (cf. Figure 5.2).

The left-hand side of Figure 5.2 shows the input product *Car Body with Dashboard* to the process *Fasten Screw & Measure*. The expected output product is *Car Body with screwed-on Dashboard*.

To realize this goal, the process is carried out by an *Electric Screwdriver* resource assembled on a *Robot*. This robot is controlled by a *Robot Controller* resource. The electric screwdriver is equipped with a *Bit* designed to carry out the screwing process. The electric screwdriver is also equipped with a *Driver* that facilitates the positioning of the screwdriver on the car. Finally, the *Screwdriver Controller* automates the positioning and general functionality of the electric screwdriver and is fueled by current coming from a *Transformer*.

The orange and blue links depict property dependencies. The blue links connect properties that depend on each other and must be reviewed if one changes. The orange links connect properties that depend on each other, and if one property changes, the value has to be propagated to the other end of the dependency link.

5.2.1 Engineering Project Stakeholders in the Use Case

To better understand the roles of the stakeholders in Figure 5.2, we describe the roles below:

Product owners oversee the engineering project. In the current use case, their responsibilities encompass prioritizing and refining the change requests and presenting them to the engineering teams.

Mechanical engineers design and optimize the mechanical components involved in the screwing process, including the fixture for holding the dashboard and the mechanisms for controlling the screwdriver's movement. Ensure the mechanical system can withstand the forces exerted during the fastening process and maintain the required positioning accuracy.

Electrical engineers design the system's electrical components, including the wiring and connections for the electric screwdriver. Implement sensors to measure torque during the screwing process, ensuring that the applied force is within specified limits. Collaborate with the automation engineer to integrate electrical components seamlessly into the control system.

Quality engineers define quality standards and specifications for the screwing process, including acceptable torque levels and positioning accuracy. Implement monitoring and inspection systems to verify that each fastening operation meets the specified quality criteria. Collaborate with all engineering domains to establish a comprehensive quality control framework for the assembly process.

Automation engineers develop the automation and control systems for the robot work cell, incorporating the necessary programming to guide the robot arm and control the electric screwdriver. Ensure precise coordination between the robot arm and the screwdriver for accurate positioning and controlled torque application. Collaborate with mechanical and electrical engineers to integrate their components into a cohesive automated system.

We omit the role of the basic planner (cf. Fig. 5.1), as it could be a person who collaborates in all domains and also prepares the first version of the concepts or has a quality engineer as a proxy [Rinker et al., 2023b].

5.3 Minimal Engineering and Change Management Process

To guide the development of the solution approach, we will work with the use case *Fasten Screw and Measure*. Figure 5.3 shows a minimal engineering and change management process focusing on multi-domain CIA based on [Rinker et al., 2022]. The figure abstracts

the initial process diagram from [Rinker et al., 2022] from the possible solution outline (mention of *Pull requests*, unified model, etc.) to enable execution of both the traditional and the proposed approach, to allow for comparable evaluation. Therefore, the thesis defines the following phases for evaluation:

Phase 1 Specification of a change: Changes to a CPPS could be initiated from multiple sources, such as technological advancement (e.g., new resources that are more efficient), market demand (e.g., innovating the output products to stay competitive could result in a change request to the existing CPPS), economic factors (e.g., substitution of existing system parts for more cost-efficient alternatives), or customer feedback (e.g. low-quality product, or request to re-design the product which implies updates to the CPPS). These requests for a change are prioritized, refined, and finally presented to the engineers by the product owner.

Phase 2 Implementation of the change request: the request is assigned to an engineer from a domain related to the assets that should be changed. The change is implemented.

Phase 3 CIA: before the engineer hands in the implemented change, the impact on the related domains and their assets should be assessed.

Phase 4 Review of impacted assets after the change: if the CIA identified any impacted system part, the impacted asset has to be reviewed by an engineer from its related domain. In Figure 5.3, the change implemented by the electrical engineer impacted assets in automation and mechanical engineering domains. The reviewing engineers must decide whether the change is acceptable and can be integrated into the common model or escalated, or they must request improvement.

Phase 5 Multi-disciplinary rework: after the change request is implemented and handed in for CIA, the engineers from impacted domains confirm that they have reviewed the initial change request and its implementation. If the change needs improvement, an engineer from the impacted domain has to conduct the rework (c.f. Figure 5.3, Mechanical engineer). If the change is acceptable, no further multi-disciplinary rework is necessary (c.f. Figure 5.3, Automation engineer). The product owner can also review how the change was implemented, who was involved, what domains and assets were impacted, and who confirmed the change's inclusion in their specific domain context.

Phase 6 Change integration: if no impacted assets were identified during the CIA, or the multi-disciplinary rework was successfully concluded, the changes to the system assets can be integrated into the common model to provide a holistic view of the system with the recent changes.

We refer to this engineering and change management process as *minimal* process, as it depicts the simplest success path from a change request to change integration. The escalation steps are not further detailed.

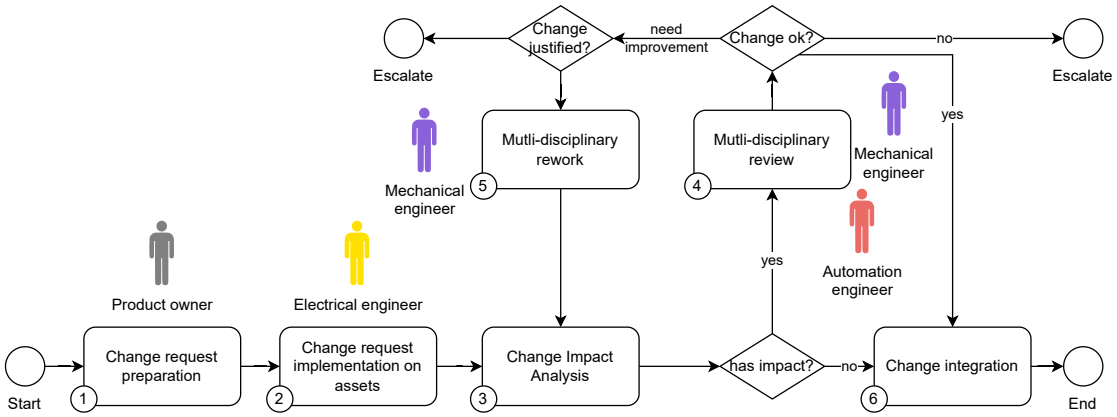

Figure 5.3: Minimal engineering and change management process based on [Rinker et al., 2022].

CHAPTER 6

Approach

This chapter covers the solution approach to tackle the defined problem and describes the planned solution elements. Section 6.1 introduces the expert survey and its results. The section also addresses the RQ1. Section 6.2 presents the M-CIA method to address the challenges of multi-domain CIA and the RQ2. Section 6.3 introduces the system design that supports the M-CIA method and addresses the RQ3. Finally, Section 6.4 introduces the prototype and shows the feasibility of the method and the system design.

6.1 Expert Survey

To elicit the state of the art in the field of CIA in the industry, this thesis introduces an expert survey with a questionnaire.

A questionnaire comprises a set of questions directed at individuals with specific demographic characteristics to gather statistically relevant information on a specific subject. Throughout the questionnaire design process, we will follow the best practices summarized by Roopa and Menta Satya [2012], as the construction of a questionnaire is pivotal for the success of a survey. Properly phrased questions, a logical sequence, appropriate scaling, and a well-organized answer format contribute to the survey's efficacy, enabling it to accurately reflect the perspectives and opinions of the participants [Roopa and Menta Satya, 2012]. To ensure the questionnaire precisely captures the intended information, a beneficial practice is to pretest it among a smaller subgroup of the target respondents [Roopa and Menta Satya, 2012].

6.1.1 Questionnaire design

The survey will be conducted online in written form and answered by experts from the manufacturing or production industry, consultants from these two industries, or researchers. The survey consists of 24 questions across three categories:

Category 1 – *Change management process in your system environment*: the category includes questions regarding how engineering changes are handled in the system environment, how dependencies are identified, questions regarding tool support of change management as well as CIA, and subjective evaluation of the ability to find the right domain that is impacted by an engineering change.

Category 2 – *Requirements for a successful and complete CIA*: This category focuses on criteria for a complete CIA, its documentation, and reporting in the context of change management. We have also presented the agile Workflow for Change Management [Rinker et al., 2022], the underlying process used in this research to understand how helpful our planned method would be to their environment.

Category 3 – *General demographic questions:* The category gathers demographic data of the respondents, such as what field they work in, the size of the company, their domain, workgroup structure, and geographic region.

We decided to ask the demographic questions at the end of the questionnaire so that the respondents would focus on the topic in the beginning and prevent leaving the questionnaire in the early stage of the survey participation.

We have developed the survey using LimeSurvey[1]. LimeSurvey is a more complex tool than Google Forms and offers more flexibility with professional questionnaires, including a custom privacy policy, more answer types, and analytic capabilities. To ensure the quality of the questionnaire, we will conduct three test rounds with friendly users to check for the understandability of questions and answers, relevance of questions, and reproducibility of the results.

For simple analysis, we create closed-ended, matrix, and contingency questions (for clarification in relevant cases) to standardize our results. In case of higher relevance or reasoning behind one's selection, we ask open-ended questions that will be analyzed manually.

6.1.2 Learnings from the Survey

This subsection presents the most interesting learnings we derive from the survey.

Demography and general questions about the organization. Ten experts from various fields and roles filled out the survey between July 2023 and August 2023. The industries of the participants' organizations are displayed in Figure 6.1.

We have formulated multiple choice answers based on the *Global Industry Classification Standard*[2]. There are four industries that the participants work in: *Materials, Capital*

[1] LimeSurvey: `https://www.limesurvey.org/`

[2] Global Industry Classification Standard: `https://www.msci.com/our-solutions/indexes/gics` and accessed on August 29, 2023.

Goods, Information Technology, and Research & Education. The latter was manually added by participants as "Others" *Materials* industry encompasses Chemicals, Construction Materials, Metals & Mining. *Capital Goods* industry encompasses aerospace & defense, construction & engineering, electrical equipment, and machinery.

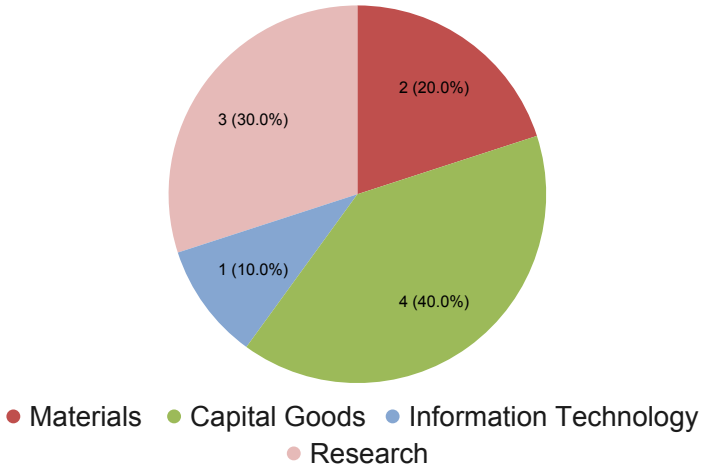

Figure 6.1: Industry of the participants' organizations based on Global Industry Classification Standard.

Geographically, 70% of the organizations of survey participants are located in Central Europe, while 20% are located in Western Europe, and the remaining 10% of the respondents are located outside of Europe. The size of the organizations that participants work in is in ratio 5:4:1 for large (3000+ employees), medium (300-3000 employees), and micro (< 30 employees) organization sizes.

We provided a taxonomy of CPPS based on Elmaraghy [2005], with the following options: *specific production systems*, *flexible production systems*, and *reconfigurable production systems*. The cyber-physical production systems were classified as flexible production systems in 30% of cases, specific production systems in 20% of cases, and 10% of the systems as reconfigurable production systems. One respondent (10% of the cases) added research demonstrators as a category in the " Others " option. The remainder did not answer the question.

Figure 6.2 shows the participants' expertise. Multiple selection was allowed to answer this question. There was often a combination of non-technical expertise, such as *management*, *data science*, or *business analytics*, with technical expertise, such as *mechanical*, *electrical*, or *process engineering*.

When it comes to the nature of work, there were 50% of the participants conduct research at an academic institution, and 50% of the participants work as an expert in an organization in the industry. Finally, all participants said they work in work groups or teams of employees from different domains (cross-functional teams).

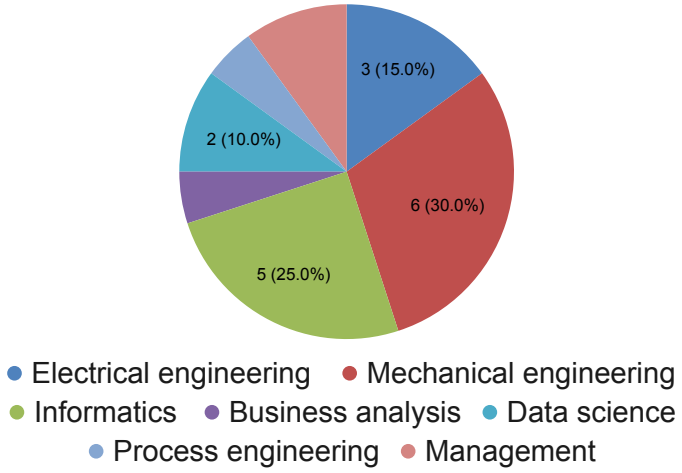

Figure 6.2: Expertise of the participants.

Change management process in your system environment. The survey showed that 70% of the respondents do not use change management tools. In contrast, only 20% of the respondents said their tool supports the identification of change impact, as shown in Figure 6.3.

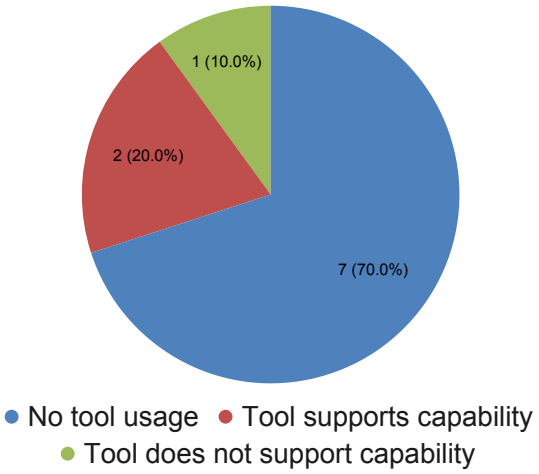

Figure 6.3: Tool support for identifying the impact of a change.

Additionally, only 10% of the respondents stated that the tools they use do support stakeholder analysis in the context of CIA (Figure 6.4).

The CIA is conducted based on previous experience and patterns in 70% of the cases. In contrast, model-driven methodology and manual analysis are conducted each in 10% of

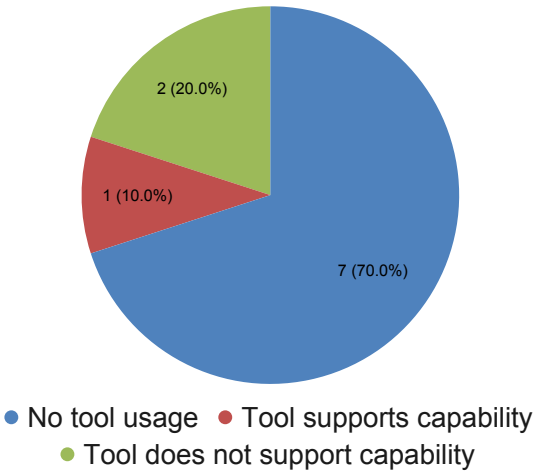

Figure 6.4: Tool support for stakeholder analysis.

the cases (Figure 6.5).

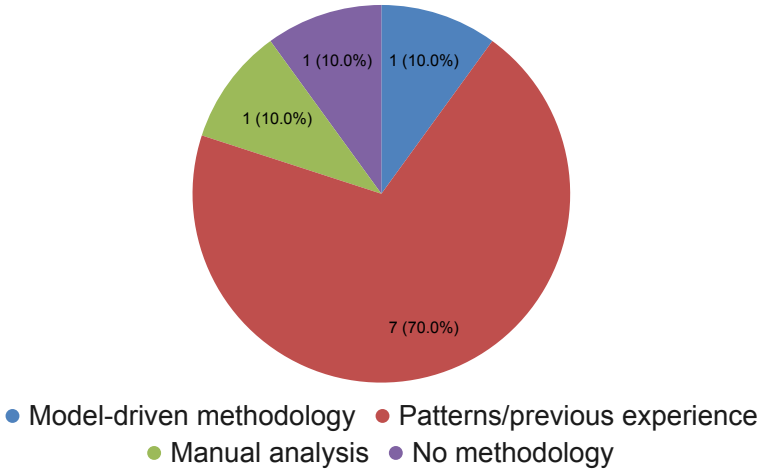

Figure 6.5: Methodology used to analyze the impact of a change on the system.

We also asked respondents to reflect on how easy it is to 1) identify technical dependencies in their system environment, 2) identify the right stakeholders from their domain who should be involved in the CIA, and 3) identify the right stakeholders from other domains who should be involved in the CIA.

Figure 6.6 shows that the respondents disagree that finding technical dependencies is easy. They tend to agree that identifying stakeholders within their domain is easy. Finally, the respondents tend to disagree that it is easy to identify the right stakeholders from other

domains who should be involved in CIA.

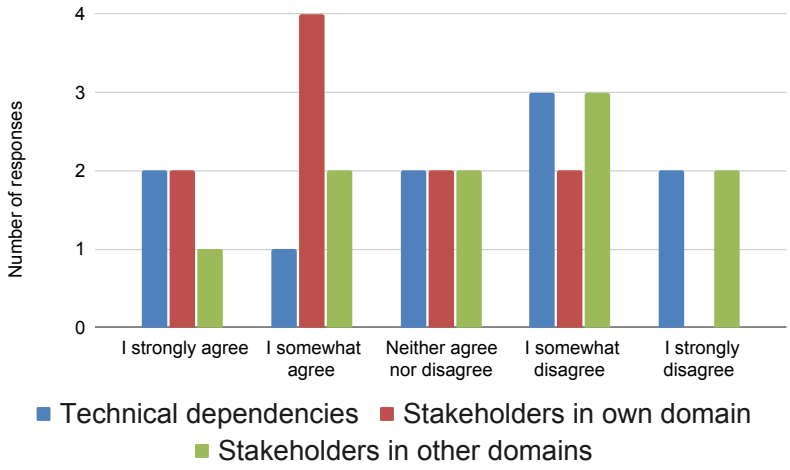

Figure 6.6: Tendency of easiness to identify the technical dependencies, stakeholders in own and other domain, in the system environment.

Following, the respondents reflected on how robust is their current CIA tool support and whether it needs improvement. We asked them to reflect on the following statements: *Our tool landscape supports CIA very well.* (c.f. Figure 6.7 "We have satisfactory CIA tool support"), and *"I wish our tool landscape would support CIA better to reduce manual effort and error-proneness."* (c.f. Figure 6.7, "We need better CIA tool support").

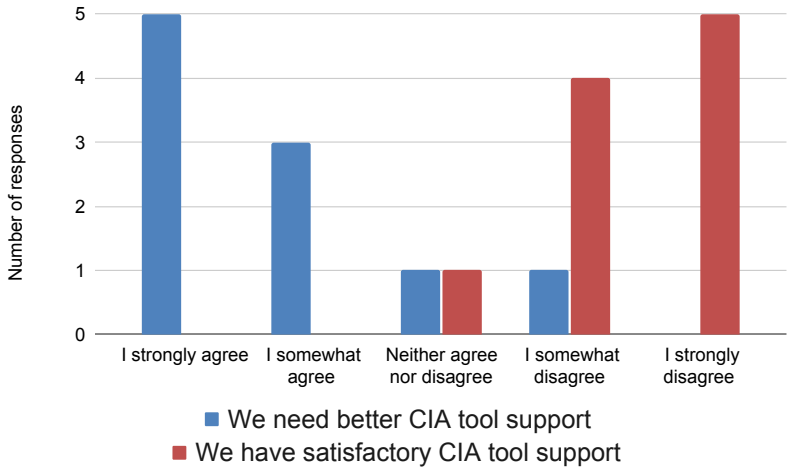

Figure 6.7: Tendency of easiness to identify the three aspects in the system environment.

We conclude that the current CIA tool support used by the respondents is not good enough for their use cases, and improvement is necessary. The respondents also reflected

on whether identifying a change's impact happens on time before it gets costly. The results are depicted in Figure 6.8. We conclude that the impact of a change is not identified on time, which results in high re-engineering and personal costs.

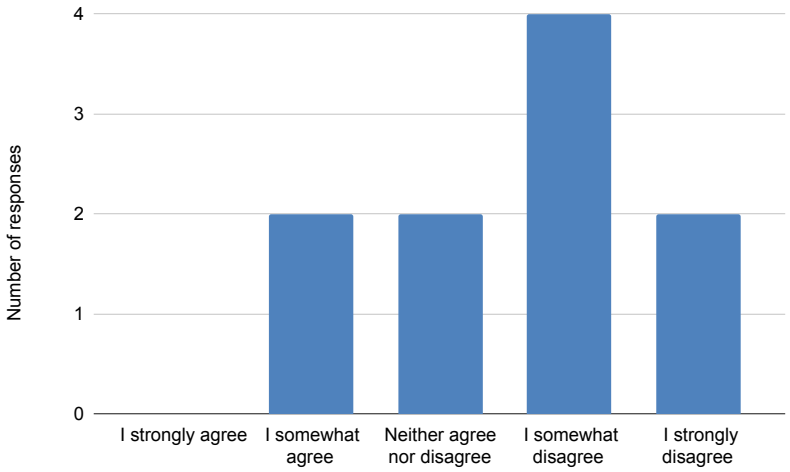

Figure 6.8: Response to the question "The change impact is identified before it gets costly".

Finally, the survey found that it is more common for engineers to deal with changes that impact system parts in other domains rather than in the source domain of change, as shown in Figure 6.9. We derive an ascending tendency of cross-domain change frequency and a descending tendency of single-domain engineering change frequency.

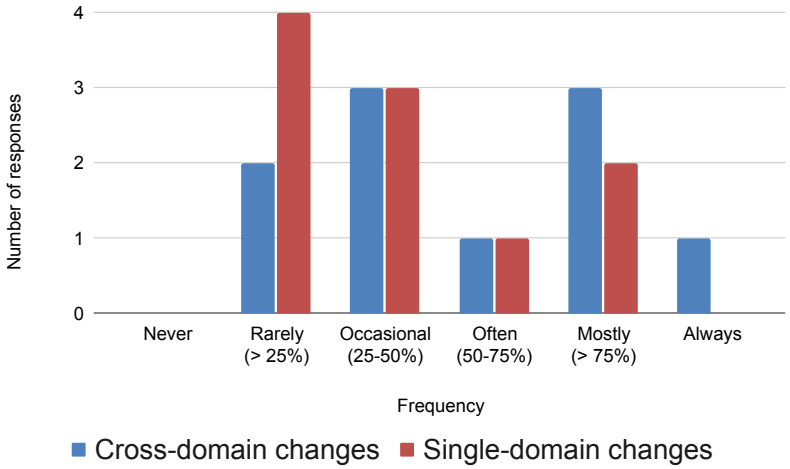

Figure 6.9: Frequency of changes with impact in other domains vs. impact in a single domain.

Requirements for a successful and complete CIA. To understand how the organizations represented by the respondents ensure that the CIA is successful and complete, the questionnaire first inquired about the approach to the documentation of CIA results (c.f. Figure 6.10). The *tool-based* option included tools such as Jira, Confluence, or Enterprise Architect. The *creation of the digital documents* option included using Microsoft Office or similar office tools. The *creation of analog documents* option represented hand sketches or drawings. We learned that the Office tools are the most used approach to document CIA results.

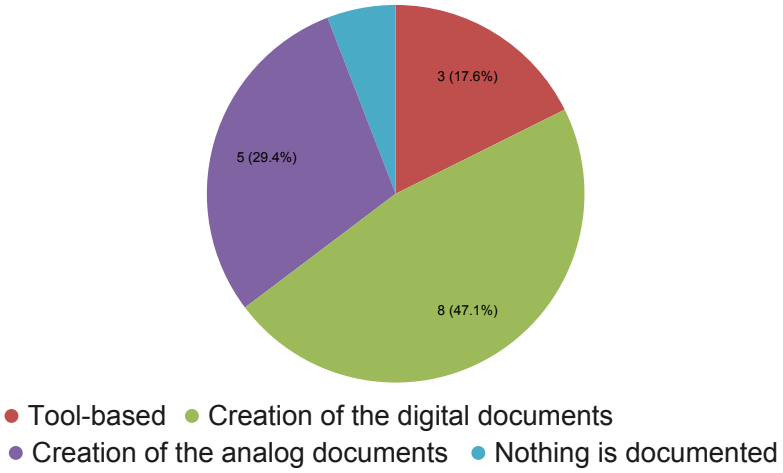

Figure 6.10: Approaches to the documentation of CIA results.

Subsequently, the questionnaire asked them about the form in which they collectively communicate the CIA results, whether or not they have any reporting in place. Only 20% of the respondents responded positively. However, 30% stated that they have a mechanism for visualizing the CIA result, e.g., success, failure, warning.

The open-ended question regarding the target group for reporting the CIA results were answered with *project manager, cluster lead (he leads a team made of stakeholders from multiple domains), client, and steering committee.*

The use of the reports was elicited via another open-ended question, which was answered with *capture cost*, *minimize impact*, *overview*, and *make management decisions.*

The last open-ended question regarding the reporting was about the content of the reports. The respondents answered the question with *a traffic light system (red - critical change, green - uncritical change), costs, impact, duration of the change implementation, who implemented the change, what was changed*, and *lead time.*

The last set of questions was focused on the Git Workflow for MvCM proposed by [Rinker et al., 2022]. We introduced the workflow briefly in the questionnaire and asked several

questions to assess whether or not the respondents could adopt the approach. The results are depicted in Figure 6.11.

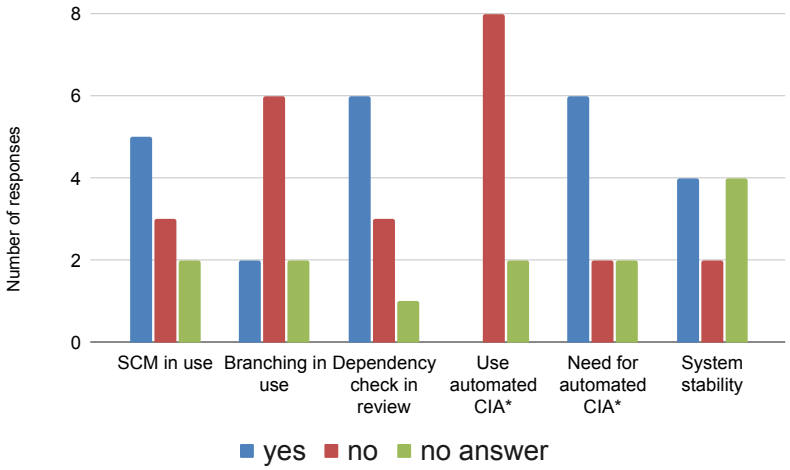

Figure 6.11: Questions regarding the adaptability of MvCM by Rinker et al. [2022].

First, MvCM would require adopting source code management such as Git. We learned that 50% respondents use Git in their organization. 20% of the respondents stated that they have a branching strategy in place, which would also be a pre-requisite for the workflow. The questionnaire also inquired whether the participants checked for dependencies and possible impacts in their current review process. This was the case in 60% of the cases.

Next, the questionnaire inquired whether or not the CIA is conducted automatically by software, and the response was *no* in 80% of the cases; the rest did not answer the question. This result would be deducible from the previous questions regarding their CIA approach; however, we added this as a consistency check. Then, the questionnaire asked the respondents whether they wished they could conduct the CIA automatically. 60% of the respondents said yes, 20% said no, and the rest did not answer the question.

Finally, the questionnaire inquired regarding the system's stability to understand whether the system concepts are immutable and whether the effort to create the necessary formal knowledge representation would pay off. 40% of the respondents said yes, 20% said no, and the rest did not answer the question. The respondents also added that such a knowledge representation might be hard to create due to the implicit knowledge of the stakeholders and hard to extract from the existing relevant documents. Additionally, they estimate high organizational and time effort.

6.1.3 Limitations

To gain as many responses as possible, the questionnaire was distributed via direct outreach to our network's industry experts, researchers, and consultants and published

on online platforms such as LinkedIn and ResearchGate. Consequently, we had limited control over who participated in the survey.

In general, we described the goal and target group on the survey's cover page, but no further suitability selection was conducted. This might partially influence the results. Additionally, the response sample was quite small, and the industry and roles of the participants were broad.

Thus, we acknowledge several threats to the validity of the results. Based on [Wohlin et al., 2012], we acknowledge a threat to the conclusion validity, given the *random heterogeneity of subjects*, as the participants' backgrounds and industries were broad. Additionally, *reliability of measures* might not be as high. As presented previously, we tested the questionnaire with friendly users. Testing was conducted only with three friendly users. Since we conducted the questionnaire with 10 participants, the results might have *low statistical power*.

Internal validity of the results might be threatened by *participant selection*. As we distributed the survey online in our professional network and reached out to our network directly, the participants who agreed to participate were volunteers. As Wohlin et al. [2012] describe, volunteers are more motivated to help gain the data and might not accurately represent the whole target group.

Finally, the construct validity of the results might be threatened by *inadequate preoperational explication of constructs*. Although we explained the constructs and terms in the questionnaire, it was not always possible to provide an extensive explanation due to the aimed brevity of the questionnaire and to ensure that all participants have the same understanding of them.

Given these threats to validity, we suggest building on the preliminary results of the questionnaire in future work and conducting a deep dive survey in the form of semi-structured interviews to foster construct validity. Besides, we suggest approaching participants at industry conferences or exhibitions to eliminate participant selection from personal networks to eliminate threats to internal validity. To foster conclusion validity, the interviews should be conducted on a bigger sample to ensure high statistical power, but at the same time, the participants' demographics should be narrowed down. Finally, the survey should be pre-tested on a relevant participant sample with the selected demographics.

6.2 M-CIA Method

This section introduces relevant requirements for multi-domain CIA. Then, the section introduces the M-CIA method, which is based on the MvCM [Rinker et al., 2022], Multi-view Model Transformation (MvMT) [Rinker et al., 2023b] and extended with learnings from the literature. The method's preliminary version and feasibility discussion were published during the thesis project [Rinker et al., 2024].

6.2.1 Requirements

The subsection defines the requirements for an efficient CIA based on the literature and expert survey. Koch et al. [2016] have derived requirements from workshops with a group of experts from the management of engineering and manufacturing changes with 10 participants. They cluster the requirements into aspects, such as *holistic view*, *transparency & traceability*, *practicability & applicability*, *process orientation*, *proactivity*, *problem-solving and analytic capabilities*, and *knowledge management*.

Rinker et al. [2022, 2023b] define the requirements for efficient change management based on the use case analysis. The requirements coming from these two papers focus on *multi-view configuration management and modeling*, *change tracing*, *change coordination*, *distributed process synchronization*, *version management*, and an efficient multi-view change management process.

There is an overlap of the requirements from the literature and the expert survey. Therefore, we combine them and define the following requirement list applicable to the use case from Chapter 5:

R1 - Holistic view: Koch et al. [2016] define this as the capability of systemic view and interfaces to other departments. We adopt this requirement as a requirement for the possibility of showing a 1) common view of the system that contains aspects from all domains and 2) a domain-specific view, which is a projection of the system model to a specific domain. This requirement can be fulfilled with MDM capabilities and multi-view configuration management Rinker et al. [2023b], in which each domain has its view of the system. If necessary, the domain's stakeholders can view the whole system via a common model.

R2 - Transparency and traceability: This aspect is defined as a transparent approach to change management and clear responsibilities [Koch et al., 2016]. It partially overlaps with the requirement of an efficient multi-view change management process [Rinker et al., 2023b] if the process transparency and clear responsibilities are provided. We define this requirement as transparency of the CIA process, the possibility of tracing change requests, reworking and implementing the change request, and traceability of data artifact change. Additionally, the responsibilities of the stakeholders should be transparent.

R3 - Practicability and applicability: In the initial publication [Koch et al., 2016], this aspect is defined as enterprise-independent applicability and simplicity of the method. We define this requirement as being applicable to both discrete and batch production environments and the simplicity of the method.

R4 - Process orientation: This aspect was initially defined as coordinating activities and stakeholders and information flow with communication support. We see an overlap with the definition of an efficient multi-view change management process, change coordination, or distributed process synchronization [Rinker et al., 2022].

We define this requirement as transparent and traceable change impact review coordination and tool support with clear steps that are easy to follow (c.f. Figure 6.3).

R5 - Proactivity: This requirement is defined as change identification and early change approval in the initial publication [Koch et al., 2016]. Version management mentioned by Rinker et al. [2023b] could also be seen as part of this aspect. Therefore, we define this requirement as semi-automated version management support, automated review coordination, stakeholder identification (to address the existing gap, c.f. Figure 6.6 and 6.4), proactive change propagation and stakeholder notification regarding the changes.

R6 - Problem-solving and analytic capabilities: Defined initially as cause and impact analysis and change classification [Koch et al., 2016]. Additional requirements from the initial publication are finding solutions and considering production system properties. We define this requirement as contextualized CIA and change review, in which the context of the change and the reasoning behind the review request is clear.

R7 - Knowledge management: Initially defined as archiving and tracking information and lessons learned [Koch et al., 2016]. The requirement of change tracing by Rinker et al. [2022] could belong to this aspect. We define this requirement as centralized documentation of the change implementation, review, and rework process. Additionally, there should be means to document new learnings, such as additional dependencies between assets to facilitate the current pattern- and experience-based approach to CIA (c.f. Figure 6.5).

6.2.2 Pre-requisites

To facilitate the change management process and multi-domain CIA, the CPPS has to be formally described to provide the data for the process. We rely on a model-driven approach, as described in Chapter 3, as it is beneficial first to make changes to the system model, validate it, and finally, perform the change on the real system. To collect the relevant system data in the form of the system model, we utilize the MDM-CPPS method's *project setup* phase, introduced in detail in Chapter 2.

The required output of the *project setup* phase is a *MDM-CPPS Project Repository* with relevant domain-specific workspaces that contain the domain *Concept Glossary* and a domain-specific system model. The repository also has to contain the common workspace, in which the *Common Concept Glossary* is defined, as well as the common system model.

In addition to the domain and system modeling activity of the *project setup* phase, it is required for the method to define the responsibilities in the given CPPS project. We define the *responsibility* as affiliation to the domain and a role (*reviewer, decision-maker*). For each of the domains (represented with domain-specific workspaces), an assignment of the stakeholders has to be created. It is allowed to assign one person to multiple

domains. In each domain, there has to be at least one stakeholder with the role *reviewer*, who is nominated to conduct the multi-disciplinary reviews. Similarly, in each domain, there has to be at least one stakeholder with the role *decision maker*. We define the role of the decision maker as a person knowledgeable about the goals and constraints of a given domain to the extent that once *escalation* is necessary (c.f. Figure 5.3, Chapter 5), this person can make decisions and provide direction to proceed in the change request resolution.

6.2.3 Multi-domain Engineering and Change Management

Once the project is set up, the engineers carry out their daily business in the *engineering and change management* phase, as described in our technical report [Rinker et al., 2024].

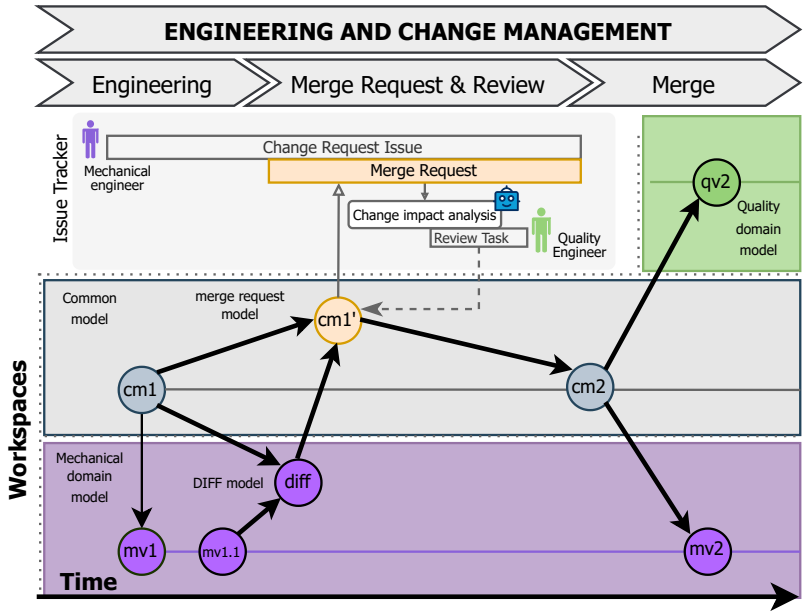

Figure 6.12: *Engineering and change management* phase as defined in our previous work [Rinker et al., 2024].

Figure 6.12 shows the *engineering and change management phase*, with the *engineering*, *merge request & review*, and *merge* activities. Additionally, we provide a mapping of the terminology to the generic methodology for *domain-spanning CIA* proposed by Heinrich et al. [2018].

Engineering

During the *engineering* phase, initiated by a *change request* detailed in an issue tracker, domain engineers work in their domain-specific workspaces, adjusting the domain-specific model to accommodate the requested modifications [Rinker et al., 2024]. Here, we suppose that the engineering teams follow a value-driven change-request-based approach, in which

changes to the model, and consequently to the system, are made based on pre-defined *change request*.

The change request is specified and prioritized by a product owner, as described in Chapter 5, based on a value-driven trigger (e.g., technology advancement, business objectives, or customer feedback). The change requests can be grouped into epics, as known from agile software engineering. To illustrate the relationship between a change request and an epic, we describe an illustrative epic and its two fictitious change requests:

Epic Dashboard Mounting Process Optimization

Change Request 1 Improved Dashboard Mounting Process. The goal is to optimize the dashboard mounting process to reduce assembly time and enhance overall efficiency. It is necessary to modify the assembly sequence in the electric screwdriver controller and to update the bit type in the electric screwdriver.

Change Request 2 Torque Control for Consistent Screw Tightness. The goal is to implement a torque control mechanism in the electric screwdriver to ensure consistent screw tightness. It is necessary to integrate torque sensors in the electric screwdriver and adjust the assembly sequence to account for torque control.

Additionally, the change request can be assigned to a milestone to align the engineering cycle to the time plan of the business. The change requests are then assigned to engineers who have the necessary knowledge to implement them. The assignment is coordinated by the product owner or agreed on in team planning.

The change request can only be assigned to one stakeholder, who is held accountable for the implementation. However, he does not have to implement everything on his own, and the creation of sub-tasks for the change requests is possible (and consequent assignment of the sub-tasks, also to stakeholders from other domains).

The change requests are displayed on the change request board of the engineering team, which consists of multiple task buckets, such as *Open, Work In Progress, Review*, and *Done*.

Finally, the system models are updated according to the change request on a dedicated change request Git branch in the domain-specific workspaces [Rinker et al., 2024]. Each of these adjustments results in a new version of the domain-specific model within their respective workspace. For example, the mechanical engineers may modify the attributes of a screwdriver (as illustrated in Figure 6.12, transitioning from *mv1* to *mv1.1* in the mechanical workspace). As the engineer proceeds to implement those changes, his domain's workspace stays untouched by activities in other domains [Rinker et al., 2024]. We achieve an isolated environment in which engineers can make their adjustments.

Merge Request & Review

The *merge request & review* activity involves several steps, with a distinction made between the initiating and affected domains. Engineers in the initiating domain aim to integrate their *seed modifications* [Heinrich et al., 2018] of the domain-specific model into the *common model* (cf. Figure 6.12, identified as *cm1*), which originates from the *project setup* phase [Rinker et al., 2024].

To facilitate the review process, we adopt a well-known concept of the pull request from software engineering, hereafter referred to as a merge request, in alignment with GitLab's terminology. This approach is derived from the MvCM [Rinker et al., 2022]. Therefore, the engineer assigned to the change request creates a merge request for the dedicated change request Git branch, on which the changes to the model were implemented. The merge request is assigned to the product owner as a reviewer for transparency and coordination reasons.

Next, a *diff model* is generated by comparing the latest version of the *common model*, also called *base model* by Heinrich et al. [2018], to the updated domain-specific model (called target model by Heinrich et al. [2018]) of the initiating domain to list the changes. Figure 6.12 labels the comparison with *diff* for the diff model, *mv1.1* for the target model, and *cm1* for the base model. This way, the domain engineers can re-check their changes with the *common model* and, if necessary, revert or rework them. The diff model is created for each domain workspace separately.

Subsequently, a *semantic CIA* is conducted for each domain participating in the project to determine if the alteration affects their local model. This requires the computation of a *change request model* (as illustrated in Figure 6.12, labeled *cm1'*) concerning the *common model* and the *diff model* (Figure 6.12, labeled *cm1* and *diff*) [Rinker et al., 2024]. This analysis is instrumental in assessing whether the concepts within the domain-specific models are impacted and is explained in the next subsection.

For the affected domains, *change review tasks* are created in the issue tracker and assigned to the engineers of the affected domains automatically using *Algorithm for Derivation of Task Lists* (c.f. Section 3.4). The engineer assignment is based on the additional information elicited in the *project setup* phase regarding the reviewer roles per domain and the capacity of the reviewers based on the count of non-closed tasks in the issue tracker.

Each review task contains a reference to the initial change request, a reference to the change request Git branch, and reasoning behind why and what they should review (c.f. Metamodel of Decision Supporting in Section 3.4). This approach provides contextualized information for improved change impact review coordination. The review tasks are displayed on the review task board, which is initially created for each domain specifically and consists of multiple task buckets, such as *Open*, *Review*, *Rework Requested*, *Rework*, *Escalate*, and *Done*. The *Duplicate Elimination Algorithm* (c.f. Section 3.4) is used to check whether a review task with the same goal has already been created to avoid the creation of duplicate review tasks.

After the merge request is created and the review tasks are created, a check for change propagation is performed. If change propagation is possible, based on the *Algorithm of Change Propagation Analysis* (c.f. Section 3.4), the change is propagated automatically to the models of other domains. This step is performed on the change request Git branch so the stakeholders from the impacted domain can review and, if necessary, revert the changes.

Merge request serves as a review platform for discussing and deciding on the approval or rejection of the implemented change [Rinker et al., 2024]. Suppose the updated model corresponds to the *impacted* domain's requirements. In that case, the review task owner closes the task and, with this, expresses their approval for merging the changes into the *common model* during the merge activity.

However, to proceed to the merge activity, all review tasks must be closed. It is the responsibility of the merge request reviewer (in our case, product owner) to keep track of it. Conversely, if the change impacts the model in another domain and rework is necessary, either the change request owner or an engineer from the impacted domain has to improve the models, taking into account the comments and decisions provided during the review process, documented in the review task of the merge request.

Merge

Once the review is concluded and all review tasks are closed, the process continues to the *merge* activity [Rinker et al., 2024]. Within the merge activity, the *change request model* (as depicted in Figure 6.12, labeled *cm1'*) is integrated into the *common model* (labeled *cm1*), resulting in the generation of a new version (labeled *cm2*). The *Git merging* functionality of two branches provides text-based merging capability. This means that if a specific line of code in a given file is changed, Git recognizes this text-based change and can overwrite the old line of code with the new one once the *merge request* is merged by a user.

However, semantic merging is not part of the Git merging functionality. For this, we follow the TMvMT approach [Rinker et al., 2023b]. Accordingly, after the Git branch is merged text-based, we provide the model merging capability, labeled *cm2* in Figure 6.12. An exemplary merge would result from changing the *torque* property of an *electric screwdriver* in the mechanical workspace. After the merge, the main branch would contain this value in the mechanical workspace or other domain-specific updates to impacted assets performed during the review (due to Git's text-based merging capability). The common model would also inherit the torque of the electric screwdriver updated [Rinker et al., 2023b]. This automatic model transformation has to be documented via Git commit message to refer to the change request for traceability reasons.

6.2.4 Multi-domain Change Impact Analysis

This subsection specifies how the semantic analysis is carried out in the context of the *Merge Request & Review* activities. This contribution was partially previously published

in [Rinker et al., 2024].

Change identification

The first step towards the *semantic CIA* is to identify changes to the assets, such as the previously mentioned screwdriver. The method compares the contents of the *common model* with the contents of the given domain-specific model. The *common model* includes all assets of given CPPS which it represents. The domain-specific models, however, only include the assets that are relevant for a given domain. Therefore, the method defines a set of assets in the *common model* as A_{cm} and a set of assets in the domain-specific model as A_d, where a_d is a member of A_d and a_{cm} is a member of set A_{cm}. The set A_d is a subset of A_{cm}. Due to the subset property of A_d to A_{cm}, it is only relevant to iterate through the elements of A_d and find a corresponding element in A_{cm}. These elements are easily recognized by their qualified name q, which is immutable and stays the same across all workspaces. Once the method finds both elements, it compares the list of attribute values.

The result of this step is a collection of changes (initial change set). The initial change set is identified as follows: If the value of an attribute in a_d defined in a domain-specific model does not correspond to the value of an attribute in a_{cm}, it means that there was a change in the domain-specific model. Therefore, we create the *diff* object with the old value, new value, q of the attribute, change type *CHANGE*, and qualified name q of a_d to the change set.

Based on this new *diff* object in the change set, the method can reconstruct the engineering activities and the initial state of the domain-specific model before the engineering activities. However, should the attribute not be present in the domain-specific or common model, this case is considered faulty since we do not allow deletion or adding attribute values to assets.

Semantic analysis

Now that the change to the model in a domain-specific workspace has been identified, a knowledge graph based on the (common) concept taxonomy has to be built. Therefore, the method requires building a knowledge graph $G = (V, E)$, where concepts, common concepts, assets, and their attributes belong to the set of vertices V, and dependency as well as functional asset relations (e.g., implements, children, parents, requires, excludes) form the graph's edges E.

We identify the impact based on the resulting initial change set from the *change identification* step and semantic dependency links defined between attributes of the assets. We differentiate between change *propagation links* and *change review links*.

We traverse the knowledge graph in *Depth-First Search* manner. The termination condition for the graph traversal is 1) we found a change review dependency, or 2) there is no more semantic link to propagate the change to another attribute.

As planned as future work in our previous publication [Rinker et al., 2024], we apply the concept of *reactive links* proposed by Raţiu et al. [2022]. An exemplary definition of a reactive link could be a link between an attribute value v_1 and an attribute value v_2 of two dependent assets. We can define the link between v_1 and v_2 to be either of type *propagation* or *revalidation*.

Additionally, we can define the relationship between the two values using boolean operators, such as $==, <, <=, >, >=$, but also to represent a value transformation operation between the two attribute values. The authors also define the directions. A relation can either be unidirectional or bidirectional.

As an extension of the *reactive links* concept, the method allows textual reasoning, which should be considered during the change dependency review, omitting the mathematical expressions.

The method designates $C_{a,v}$ as a collection of changes derived from the *change identification* phase. This set comprises pairs p denoting an asset a and an attribute value v, represented as $p = (a, v)$. For each modified attribute value v and its corresponding parent asset a, the method navigates the knowledge graph to identify the dependency relation R, wherein v is either the starting or ending node of the dependency relation. Subsequently, the resulting set of impacted assets and their attribute values, denoted as $I_{a,v}$, is compiled.

Semantic relation handling

In the *semantic analysis* phase, the method described how it uses dependency relations between assets to identify impact. In the *realization* phase of the *project setup* process, a definition of relations on the concept and common concept level was proposed Rinker et al. [2024]. Later in the process, to enable semantic analysis, the method transforms the concept and common concept relations into asset relations and adds them to the *common model*.

The method defines two types of semantic links: 1) *Common Model Semantic Links,* that are allowed to be defined in the common model, and 2) *(Common) Concepts Semantic Links*, that can be defined in *Common Concept* and *Concept Glossaries*.

The method uses the following rules (cf. notations shown in Table 6.1) to transform the *(Common) Concept Semantic Links* to *Common Model Semantic Link* to interpret (common) concepts dependencies on the asset level of the *common model*, as we proposed in our previous work [Rinker et al., 2024]:

Link between common concepts, and attributes of the concepts they inhabit:
Given a relation R between Common Concept cc_{from} and Common Concept cc_{to}, for both common concepts, find the set of assets ACC_{from} and ACC_{to} that represent the starting and ending common concepts defined in the relation R. Then, compute a cartesian product of ACC_{from} and ACC_{to}. For each pair p = (acc_{from}, acc_{to}),

Symbol	Description
R	Semantic Dependency Relation
cc_{from}	Common Concept as a starting node of a relation
cc_{to}	Common Concept as an ending node of a relation
CC	set of Common Concepts
ACC_{from}	Set of assets that represent a Common Concept defined as a starting node of a relation
ACC_{to}	Set of assets that represent a Common Concept defined as an ending node of a relation
c_{from}	Concept as a starting node of a relation
c_{to}	Concept as an ending node of a relation

Table 6.1: Formal notations as defined in [Rinker et al., 2023a]

where acc_{from} owns the attribute defined as starting attribute of R and acc_{to} owns the attribute defined as ending attribute of R, create *Common Model Semantic Links* between the elements of the eligible pairs of the cartesian product. It could also be the case that there exists a pair p = (acc_{from}, acc_{to}), where either acc_{from} does not own the attribute defined as the starting attribute of R or acc_{to} does not own the attribute defined as ending attribute of R (also possibly both do not own the attribute). This could be the case if the common concept is represented by an asset that exists in multiple domain-specific models, therefore having attributes from multiple domain-specific views that are not part of the relation definition.

Link between concepts and attributes they own: For this type of link, proceed similarly. The only difference is a pre-processing step, in which we find the set of common concepts CC_{from} that inhabit the c_{from} and the set of common concepts CC_{to} that inhabit c_{to} in R. Subsequently, create a cartesian product of CC_{from} x CC_{to}. For each of the resulting pairs p = (cc_{from}, cc_{to}) of the cartesian product, find a set of assets that represent a cc_{from} and cc_{to}. The method defines these sets as ACC_{from} and ACC_{to}. Finally, we proceed with the same process steps as for the links between common concepts.

6.2.5 M-CIA Method – CIA Bot

The previous subsection introduced the building blocks of the CIA method. To visualize the method, Figure 6.13 presents activity flows of a change implementation and CIA. Figure 6.14 presents activity flows of an impact review and change request closure.

The method envisions three main entities that interact. One entity is a CPPS engineer that makes changes to the model. Then, an information system for source-code management is envisioned, with which the engineer interacts (cf. GitLab in Figure 6.13).

Finally, the CIA Bot comes into place, which observes the actions of the engineers in the project repository (in the information system) and performs actions to coordinate M-CIA.

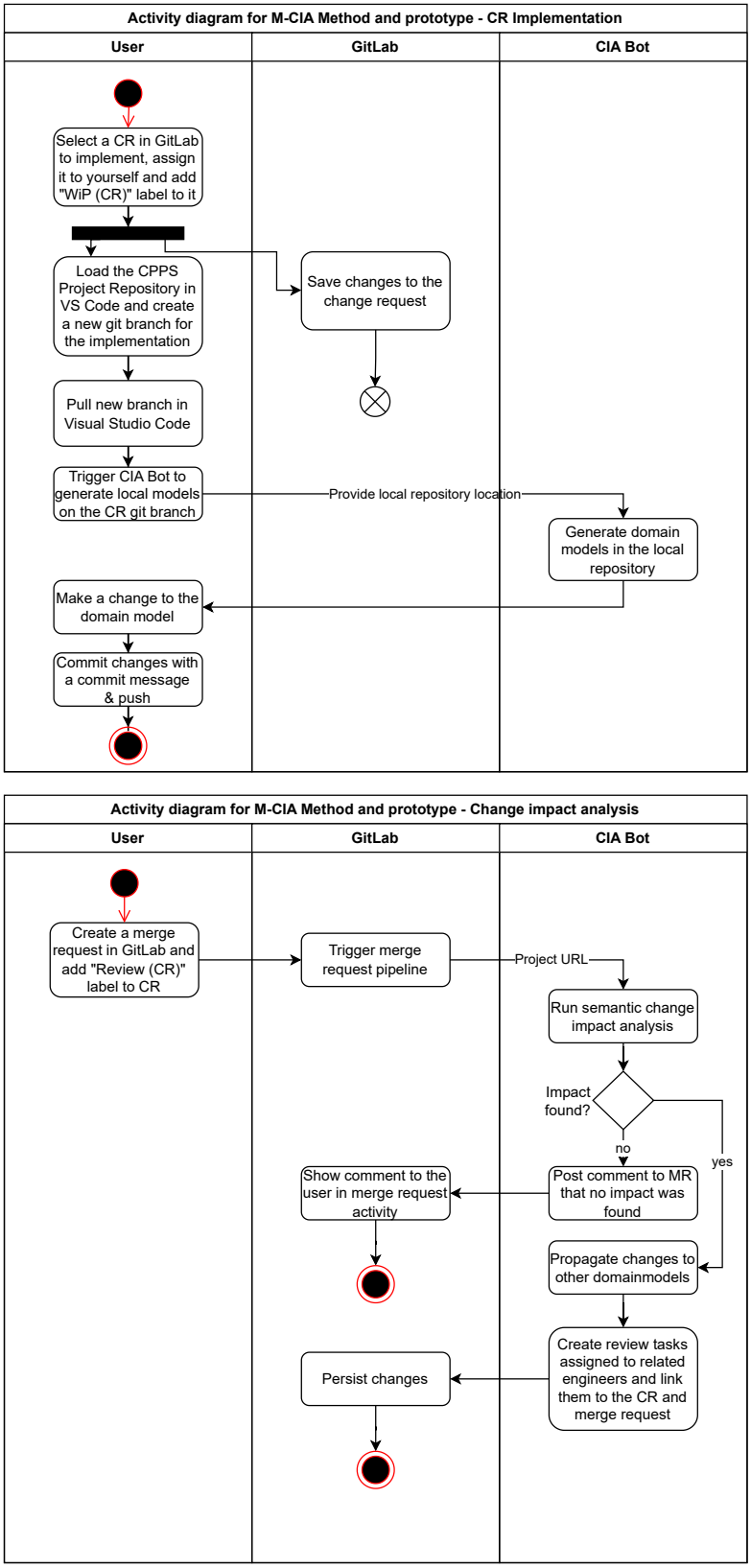

Figure 6.13: Activity diagrams for the change request implementation and M-CIA.

As illustrated in Section 2.2, this thesis contributes a definition of how to establish the necessary knowledge for automatic M-CIA coordination and notification of the CPPS stakeholders to the MvCM approach Rinker et al. [2022].

Conceptually, the *CIA Bot* is the central element that understands the changes made by the engineers either within the project Git repository or within the information system (cf. GitLab in Figure 6.13). The *CIA Bot* responses by relevant actions back in the information system or the project Git repository. In the course of the change implementation, the main role of the *CIA Bot* is to generate the domain-specific models from the common model on the corresponding Git branch in the Git repository (c.f. Figure 6.13, *CR Implementation*).

During the multi-domain CIA, the *CIA Bot* clones the project Git repository to its local file system for further semantic processing. The generated information flows back to the information system (cf. GitLab in Figure 6.14, *Impact analysis*) to either inform the user about performed change propagation in the project Git repository to other domain-specific models or to notify the engineers about necessary actions via automatically created, context-based, review tasks (c.f. Figure 6.13). The *CIA Bot* also observes the work capacity of the CPPS stakeholders to make an informed decision when assigning the review.

During the *Impact review*, the *CIA Bot* observes the actions of the reviewers. Correspondingly, the *CIA Bot* notifies the merge request owner and reviewer about the review progress. If the review results in a rework action, the *CIA Bot* executes another round of the M-CIA *Impact review* cycle (c.f. Figure 6.14, *Impact review*).

Finally, during the change request closure (cf. Figure 6.14 *Change request closure*), the *CIA Bot* task is to carry out the model transformation to a set of integrate domain-specific changes to the common model and commit these changes to the project Git repository.

6.3 M-CIA System Design

To facilitate the method execution, this section presents the IS design, called *M-CIA Management System*, to automate the M-CIA method described in the previous section. The proposed system design is depicted in Figure 6.15 based on Rinker et al. [2024] and contains three main components: 1) the MDM-CPPS Framework, 2) the IDE Ecosystem, and 3) the GitLab Ecosystem.

Parts of the system design are an advancement of the Multi-view Modeling Framework (MvMF) proposed by Rinker et al. [2023b], which enables multi-domain knowledge modeling and supports cross-domain model integration. The overall system design is inspired by the Eclipse Modeling Framework (EMF) Bruneliere et al. [2015]. However, EMF is strongly coupled with the Eclipse ecosystem [3] that hinders a direct implementation

[3]Eclipse: `https://www.eclipse.org`

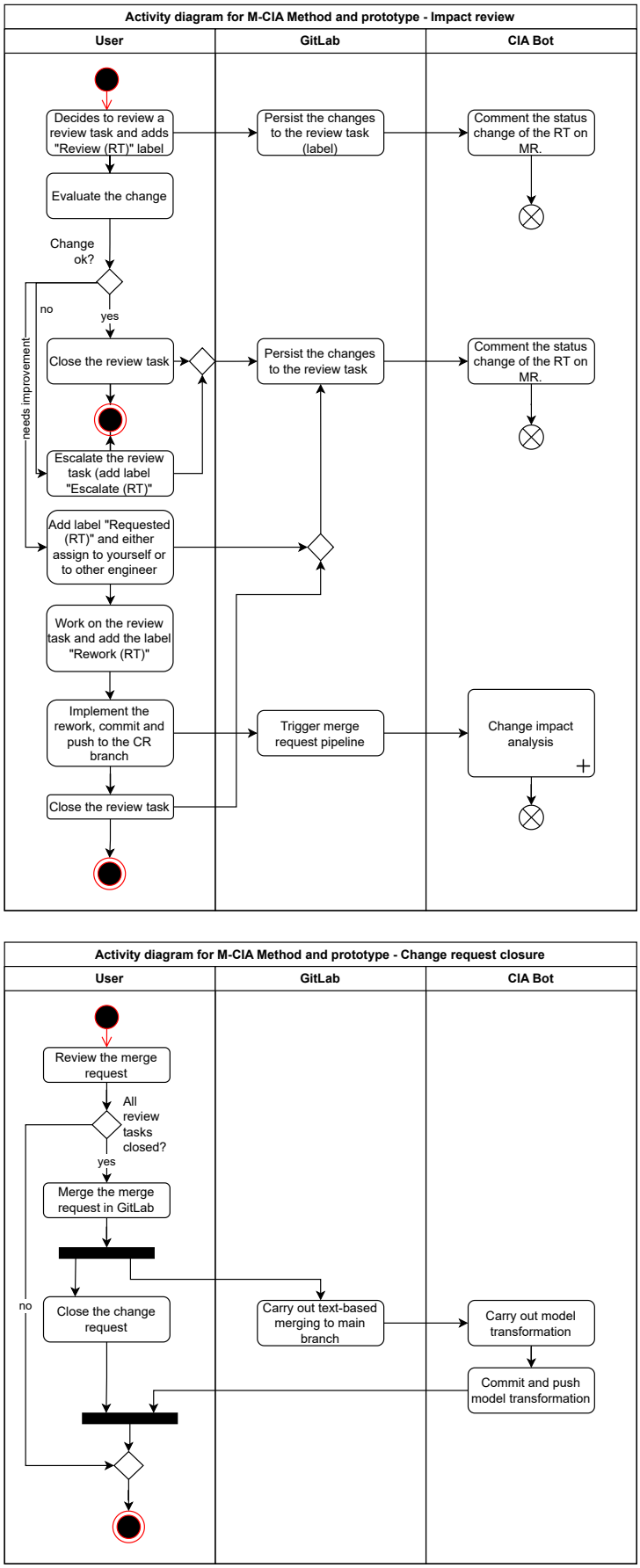

Figure 6.14: Activity diagrams for the impact review and change request closure.

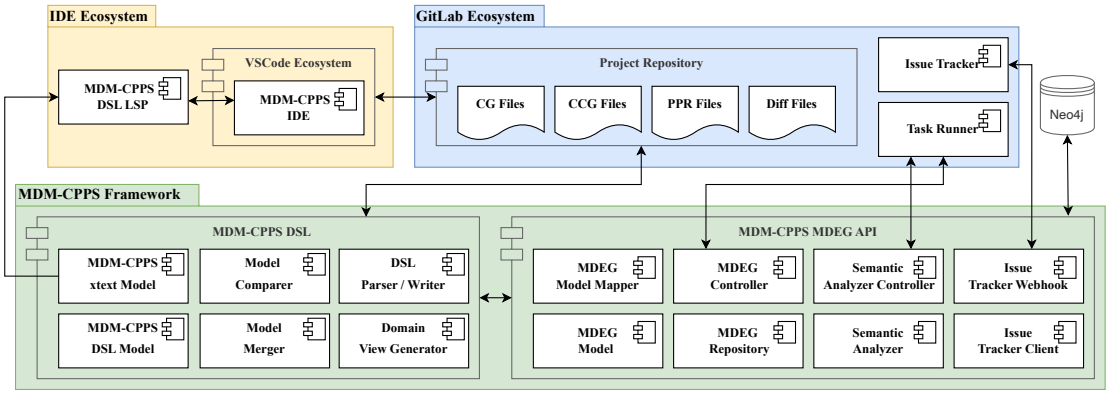

Figure 6.15: System Design of the *M-CIA Management System* based on Rinker et al. [2024].

with the framework and requires a custom software solution as stated by Batory and Altoyan [2020].

For this reason, this work builds on the TMvMT framework Rinker et al. [2023b], which proposes an Eclipse-independent system design. The *MDM-CPPS IDE*[4] is utilized and extended to enable a Git-based multi-domain workspace environment. Further, the *MDM-CPPS DSL LSP*, *MDM-CPPS DSL Model*, and *MDM-CPPS xtext Model* implementations are significantly extended, as reported in our previous publication Rinker et al. [2024], to enable the *M-CIA Management System* design of this work. In the following, the components of the designed *M-CIA Management System* are described in detail:

6.3.1 IDE Ecosystem

The IDE Ecosystem contains a source-code editor providing code intelligent capabilities. The IDE Ecosystem is not a primary contribution of this thesis, but it is vital for the definition of the system models, and its underlying source code is extended in MDM-CPPS Framework. In this case, the *Visual Studio Code*[5] ecosystem is utilized as an IDE. The main feature is the integration of the language server protocol implements Language Server Protocol (LSP)[6] to provide extended code intelligent support capabilities for custom DSL implementations. Figure 6.15 depcits the MDM-CPPS DSL LSP and MDM-CPPS IDE components.

Overall, the engineering design workflow can be defined as follows: The engineers define models of the CPPS system, using the syntax of the *MDM-CPPS DSL* in the source-code editor *VSCode* extension. Next, the *VSCode Ecosystem* provides the integration of the *MDM-CPPS DSL* into the *VSCode*, via the *MDM-CPPS DSL LSP* which implements the

[4]MDM-CPPS IDE: `https://marketplace.visualstudio.com/items?itemName=ModelIEE.mdmcpps-ide`
[5]VSCode: `https://code.visualstudio.com`
[6]LSP Protocol: `https://microsoft.github.io/language-server-protocol`

LSP for the *MDM-CPPS xtext Model*. Such integration provides syntax validation and completion capabilities in the MDM-CPPS IDE. This ecosystem is used in the system design as a support to create system models using the MDM-CPPS DSL.

6.3.2 MDM-CPPS Framework

The MDM-CPPS Framework provides functionality for describing multi-domain CPPS projects using the MDM-CPPS DSL and for graph-based analyses and change management using the MDM-CPPS MDEG API [Rinker et al., 2024]. The MDM-CPPS MDEG API is the contribution of this thesis.

In the following, the MDM-CPPS DSL component is described in detail as it is the foundation for the MDM-CPPS MDEG API:

MDM xtext Model: The xtext grammar definition is an extension of the PPR DSL grammar definition [Meixner et al., 2021b]. The initial PPR DSL was extended by the domain-specific concept and common concept modeling capabilities to realize the CCG approach Rinker et al. [2019]. This addition to the PPR asset, attribute, and relation modeling was realized in a previous implementation project of the MDM-CPPS IDE.

MDM-CPPS DSL Model: The MDM-CPPS DSL model extends the initial PPR DSL [Meixner et al., 2021b] data model with domain-specific concept and common concept modeling entities [Rinker et al., 2019]. In the course of this thesis, the MDM-CPPS DSL model was extended by glossary identification and versioning aspects to facilitate Git-based multi-domain workspace needs.

DSL Parser / Writer: To parse the *MDM-CPPS DSL* files written by the engineers in the *IDE Ecosystem*, the initial PPR DSL Parser was extended as a contribution of the thesis. The Parser is realized by *Antlr Parser Generator*[7] that is based on the previously described *MDM xtext Model*, that formalizes the syntax of the domain-specific language. The DSL writer is a custom solution, implemented as a contribution of this thesis, that prints the *MDM-CPPS DSL Model* to domain-specific files in the domain workspace.

Model Comparer: The comparer builds on the TMvMT approach from Rinker et al. [2023b]. As a contribution of this thesis, it clones the *Project Repository*, containing the common and domain-specific workspaces, to the local file system. To identify changes made in a domain-specific model compared to the common model, the *DSL Parser* parses the domain-specific view file and the common model in two instances of a *MDM-CPPS DSL Model*. Finally, the comparer compares the domain-specific model against the common model and returns a collection of changes as a *Diff file*.

[7] Antlr Parser Generator: `https://www.antlr.org`

Model Merger: The merger builds on the TMvMT approach from Rinker et al. [2023b]. In a contribution of the thesis, changes from the *diff model*, coming from the local domain workspaces, are compared to the common model.

Domain View Generator: To realize the domain-specific workspaces, the domain-specific PPR files are generated with the *Domain View Generator* from the common model. The *Domain View Generator* builds on the TMvMT approach from Rinker et al. [2023b]. As a contribution of the thesis, the *Domain View Generator* reads the concept, common concept, and the PPR assets from the common model files using *DSL Parser* and generates domain-specific model files based on the *MDM-CPPS DSL Model* using the *DSL Writer*. The domain-specific PPR models only contain PPR assets relevant to the specific domain.

The semantic analysis of the system model defined in the *MDM-CPPS DSL* files, according to the [Rinker et al., 2022, 2023b] approaches, is a core contribution of this thesis. Below, we describe the components of MDM-CPPS MDEG API, which relies on the approaches proposed by Rinker et al. [2022, 2023b], that enables further semantic analysis of the system model defined in the *MDM-CPPS DSL* files:

MDEG Model: To efficiently save the system model in a database, such as Neo4j, MDEG model [Rinker et al., 2021] contains entities to build a multi-domain engineering graph in Neo4j with appropriate relation references and database-related annotations.

MDEG Model Mapper: To represent the data described in *MDM-CPPS DSL* in a knowledge graph, we map the DSL domain model *(MDM-CPPS DSL Model)* to the graph domain model *(MDEG Model)* using the concepts of the TMvMT approach [Rinker et al., 2023b].

MDEG Controller: The MDEG controller enables the creation of an MDEG knowledge graph based on the MDM-CPPS DSL files and the graph's deletion.

MDEG Repository: To establish a connection to the graph database and to execute queries on it, a data access layer is necessary. The repository represents such a data access layer.

Semantic Analyzer: To automated M-CIA process, the analyzer uses the *Model Comparer* of the component MDM-CPPS DSL to identify the changes in domain-specific models compared to the common model [Rinker et al., 2022]. For each element of the identified change set, the analyzer traverses the knowledge graph and looks for impacted assets and their attributes. Finally, the analyzer returns the change object and its corresponding change impact information. Optionally, the analyzer triggers the creation of the review tasks for the impacted assets in the issue tracker via *Issue Tracker Client*. When called as part of the *Merge Request Pipeline*, the analyzer also propagates the value changes as specified in the knowledge graph.

Semantic Analyzer Controller: To integrate and automate the semantic analysis, this semantic analysis capability is exposed via this controller - the controller calls the *Semantic Analyzer* once it gets the request.

Issue Tracker Client: The client provides the interaction capabilities to our system to perform read and write operations on the issues and the users. Most importantly, it is called by *Semantic Analyzer* to create and manage review tasks in the issue tracker. Secondly, we use the client to check for review task duplicates to prevent redundant task creation and retrieve the work capacity of the users for the stakeholder assignment. As a solution, we use GitLab[8] as an open-source issue tracker, as it provides REST API to interact with it, but other alternatives, such as Atlassian Jira[9] would work as well. Once the review tasks are created, the client links them to the initial change request task and to the merge request to ensure change traceability.

Issue Tracker Webhook: Part of the system design relies on asynchronous communication via webhooks. To propagate the updates to the review tasks to our backend, we expose an *issue webhook* that is triggered by the issue events in GitLab. This capability is important for tracking the status of the review tasks by our backend by posting comments with the new status of the review task under the merge request. To propagate the merge request updates to our backend, we expose a *merge request webhook*, which GitLab triggers on merge request events. This is important for the semantic integration of the domain-specific models to the common model, as the integration is executed with *Model Merger* after the merge request is merged.

6.3.3 GitLab Ecosystem

To implement the proposed solution, a system ecosystem that supports Git-based repositories and issue tracking is required, in our case, GitLab. It is also necessary that the tool provides APIs to interact with it. GitLab enables us to create merge requests that trigger the merge request pipeline with various tasks running on the source files. These pipelines are executed by the *Task Runner*.

Below, we describe its three components:

Project Repository: The project is structured into a common workspace folder and domain workspace folders. Each domain workspace contains *Concept Glossary* files with concepts and domain-specific view PPR files. Additionally, the subcomponent *Model Comparer* creates a temporary *diff file* in each domain workspace, with identified changes compared to the common model. The common workspace contains *Common Concept Glossary* file with common concepts and a common model PPR file. Additionally, the project repository contains a config file in which the domains, users, and their roles are defined.

[8] GitLab: `https://gitlab.com`
[9] Atlassian Jira Software: `https://www.atlassian.com/software/jira`

Issue Tracker: To manage the change requests, requirements, and review tasks, we use the *Issue* module of GitLab and define these artifacts as issues in GitLab. Additionally, we create *Issue Boards* for *Change Requests* and an *Issue Board* for each of the domains on which the *Review Tasks* are managed. To manage the status of the *Issues*, we use labels. In GitLab, a dedicated Git branch can be created for each issue. To integrate the Git branch into the main branch, GitLab provides a special mechanism called *Merge Request*.

Task Runner - Merge Request Pipeline: Once a *Merge Request* is created to merge domain-specific model changes to the common model, the GitLab *Merge Request Pipeline* is run to create a fresh knowledge graph using *MDEG Controller* and then calls *Semantic Analyzer Controller*, which analyzes the impact of the changes in domain-specific model files and finally creates review tasks in the issue tracker or propagates the value changes as specified in the knowledge graph.

6.4 M-CIA Management System Prototype

This section introduces the *M-CIA Management System* prototype and its architecture. The system design and the method of implementation are described in detail to evaluate the feasibility of the prototype. The prototype's architecture is depicted in Figure 6.16. The architecture depicts three central components, the GitLab Ecosystem, MDM-CPPS Framework (c.f. CIA Bot in Figure 6.13 and Figure 6.14) and the Neo4j database.

GitLab Ecosystem is a collection of its sub-elements, such as *GUI*, with which the engineering teams interact; the *project Git Repository*, in which the *CPPS workspace* is stored; *GitLab APIs* which allows interaction with GitLab with HTTP(s) calls instead of the GUI, and *GitLab webhooks*, which trigger our endpoints if an issue or merge request changes; and finally *GitLab runner*[10] which is not part of the GitLab Community Edition Docker image[11] so we had to set up additionally, in order to run *Merge Request Pipeline*.

The *Merge Request Pipeline* is a mechanism for executing a set of bash commands once the merge request is created or there is a change to the code connected to the merge request. The GitLab Ecosystem (community edition) is easy to set up in a Docker[12] container.

Neo4j database is also set up locally (community edition) and runs in a Docker container to represent the MDEG of a CPPS.

MDM-CPPS Framework is a custom *Java*[13] application written with *Spring Boot*[14]. The framework follows the Service-oriented architecture (SOA) [Erickson and Siau, 2009] paradigm and uses RESTful Application Programming Interfaces (APIs). *Spring Boot* is

[10]GitLab runner: `https://docs.gitlab.com/runner/install/docker.html`
[11]GitLab: `https://docs.gitlab.com/ce/install/docker.html`
[12]Docker: `https://www.docker.com/`
[13]Java 17: `https://openjdk.org/projects/jdk/17/`
[14]Spring Boot: `https://spring.io/projects/spring-boot/`

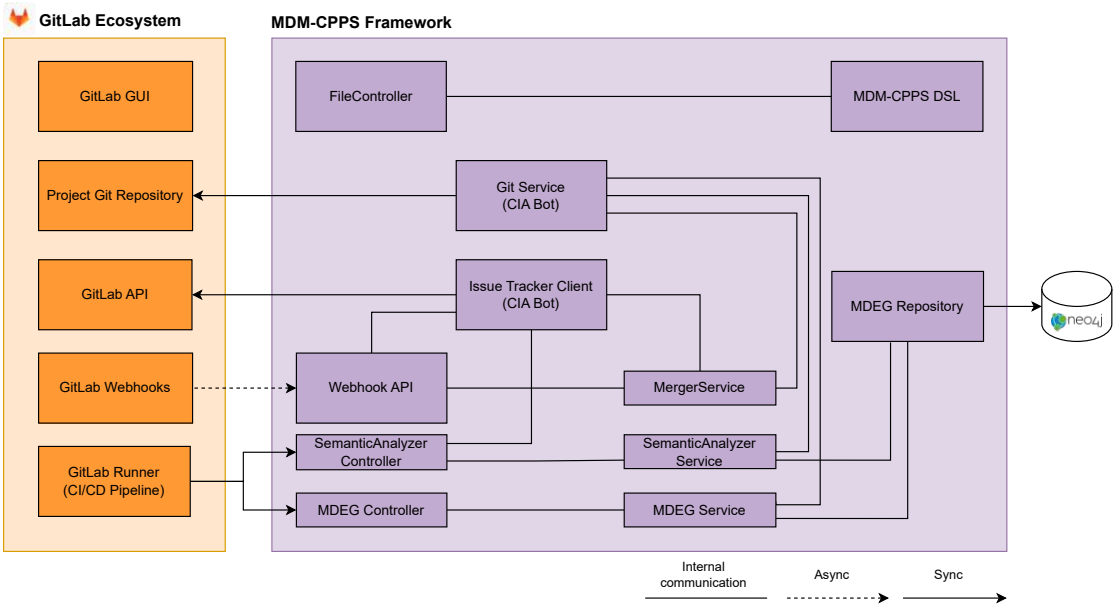

Figure 6.16: System Architecture for the *M-CIA Management System* prototype.

the state-of-the-art framework for API development using *Java* programming language. APIs are structured based on the three-tiered architecture [Tie et al., 2011] to the data layer, logic layer, and presentation layer. In *Sprint Boot* terms, the data layer is represented by the actual database and the persistence logic (also known as *@Repository*). The logic layer is represented with decoupled services (also known as *@Service*). Finally, the presentation layer in *Spring Boot* applications is represented by decoupled *Controller* classes that are exposed to the client systems, handle the HTTP(s) requests, call the application logic, and respond with the corresponding response to the client.

The MDM-CPPS Framework is visible to the user as a *CIA Bot*, as the underlying Java app authenticates in GitLab with its access token. GitLab recognizes the token as *CIA Bot* user. Therefore, actions such as creating review tasks, commenting on merge requests, assigning engineers to review tasks, propagating value changes to other domains, and merging the domain model to the system model seem for the user to be performed by the *CIA Bot* (c.f. Figure 6.13 and Figure 6.14). The functionality of the *MDM-CPPS Framework* is already described in Section 6.3. IDE Ecosystem is omitted in the system prototype architecture, as it is not a primary contribution of this thesis, and it is already explained in Section 6.3.

6.4.1 Feasibility of the Solution on the Illustrative Use Case

This section executes the M-CIA method throughout the *project setup* and *engineering and change management* phase on the illustrative use case, as previously published in [Rinker et al., 2024]. This section extends the feasibility discussion for aspects that were

not yet implemented at the time of writing the initial publication.

Project Setup Phase

In the *project setup phase*, practitioners gather the knowledge in the CPPS project. First, they define domain-specific concepts and their dependencies and consolidate them into common concepts. If necessary, the practitioners also define dependencies between common concepts. Then, a common model is created manually based on the common concepts. Finally, cross-domain dependencies are defined between assets and their attributes in the common model.

For the details of the concept and common concept definition, as well as the creation of the common model, refer to Section 2.1. This section omits these details, as the *project setup phase* is not the primary contribution of the thesis. The preliminary feasibility evaluation of this phase was published in our previous publication [Rinker et al., 2024].

However, this section discusses additions to the *project setup phase* that were proposed as part of the thesis. Concrete extensions of the relation definition and domain-specific model generation are introduced below.

Relation definition. As described in our previous work [Rinker et al., 2024], each of the *Concepts Glossary*, *Common Concepts Glossary* and *Common Model* files can contain relations. A Relation is defined by a unique *ID* and field *from*, which indicates the starting node of the relation. The field value is a combination of an asset and its attribute connected with an arrow (->). The relation is also defined by the field `to`, which describes the end node of the relation and consists of a combination of an asset and its attribute specification. The `definition` field describes a mapping between two attribute values.

```
1  Relation relation_torque {
2      from: electric_screwdriver -> MechanicalConcepts.torque
3      to: electric_screwdriver -> QualityConcepts.req_torque
4      definition: "P:=:BI"
5  }
```

Listing 4: PPR *Relation* definition for the `dependency` between the quality and the mechanical attribute *torque*, which depicts a propagation (P) with an identity function (=) in both directions (BI).

Listing 4 shows the relation with id *relation_torque*. The `from` field links to the *common model* asset *electric_screwdriver* mechanical attribute *torque*. The `to` field links to the *common model* asset *electric_screwdriver* quality attribute *req_torque*. The example shows that the two attributes that are dependent on each other.

The definition specifies a propagation dependency (P) with an identity function (=) in both directions (BI - bidirectional). Alternatively, the definition could specify review

dependency (R), either with a mathematical function of what constraint should hold or with a textual definition that has to be reviewed once one of the relation nodes is changed. The value mapping functionality was originally only conceptually defined in our previous work [Rinker et al., 2024], but this thesis realizes the links based on the reactive links concept Raţiu et al. [2022].

Domain-specific model generation. The domain-specific models are projections of the common model to the specific domain and contain only assets relevant to the stakeholder from a given domain. The changes to the model can only be made through the domain-specific models. For this, *domain-specific model* files have to be generated. Domain-specific model files contain a comment generated based on the *common model*.

Furthermore, the standard header is included. The standard header automatically increments the version number based on the version number of the source common model. The file contains all domain-specific assets and relations inherited from *(Common) Concept Glossaries* and the common model.

As defined in our previous work [Rinker et al., 2024], the PPR assets are inherited into a local file with its full qualified name (e.g. *PositionScrewDashboard_Model.electric_screwdriver* cf. Listing 5) including the field `represents` and a list of domain-relevant attributes. Additionally, optional fields, such as `children`, `parent`, `excludes`, `implements`, `requires` can be defined.

We ignore the *name* as we do not allow change of this field in domain-specific views. The assets in the domain-specific model inherit the attribute value from the asset attributes in the common model. If the attribute value in the common model is missing, the asset inherits the attribute's default value from the *Concept Glossary*.

```
1  // Generated from 'FastenScrewDashboard_Model' with version 1.1.0
2   ID MechanicalView {
3          name: "Generated Mechanical View based on 'FastenScrewDashboard_Model'"
4          version: 0.1.0
5  }
6  Resource FastenScrewDashboard_Model.electric_screwdriver {
7          name: "Electric Screwdriver"
8          represents: CommonConceptGlossary.cc_electric_screwdriver
9          children: FastenScrewDashboard_Model.bit
10         parents: FastenScrewDashboard_Model.robot
11         MechanicalConcepts.torque: 10.0
12 }
```

Listing 5: Mechanical *domain-specific model* with `Resource` *Electric Screwdriver* with mechanical attribute *torque*, as a result of the domain-specific model generation, as previously shown in our work [Rinker et al., 2024].

In Listing 5, we show a generated domain-specific model for the mechanical domain. A mechanical engineer changes the domain-specific attribute mechanical *torque* to the value *10.0*, in the *cc_electric_screwdriver*.

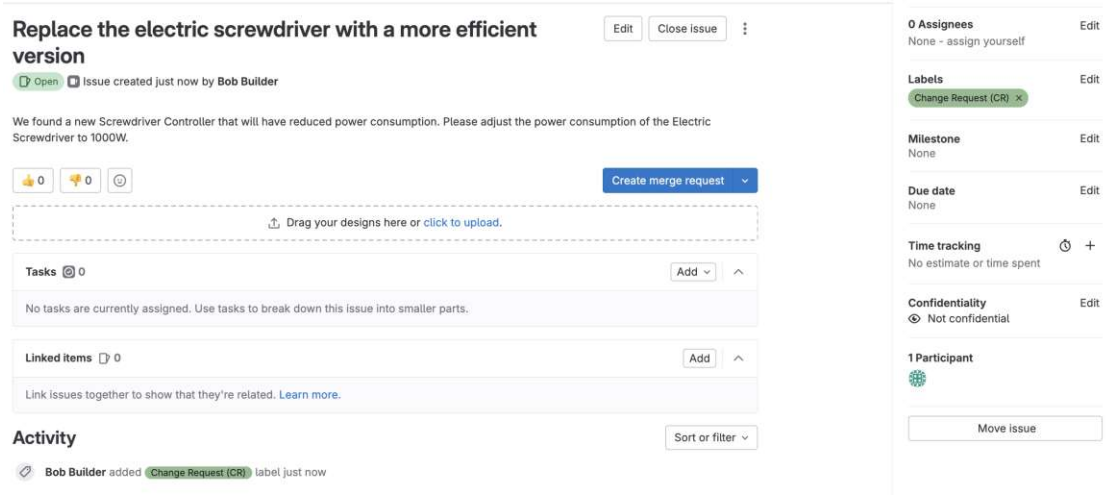

Figure 6.17: An exemplary change request.

The relations are often specified between concepts or assets of two different domains. We use the following logic to inherit them to domain-specific model files, as defined in [Rinker et al., 2024]: *"Given a relation R with two assets A_{from} and A_{to} and their attributes p_1 and p_2. If both attributes are defined in the domain concepts file, then inherit this relation to the domain-specific view file. If at least one of the attributes is not defined in the domain concept file, the relation is not inherited."*

6.4.2 Engineering & Change Management Phase

To demonstrate the M-CIA method in the *engineering and change management phase*, this subsection defines a fictitious change request. The requested change is to update the value of the power consumption property of the electric screwdriver from 1.000W to 1000.0W (cf. Figure 6.17). The exemplary change request has a short title and a description. Additionally, it has a label to represent that this GitLab issue is a *change request*.

The attribute value is changed in the mechanical domain-specific model (e.g., using the *MDM-CPPS IDE* and based on the change request documented in *GitLab*). The source files are stored in the *project repository* in GitLab (c.f. Figure 6.18).

Figure 6.18 depicts the most important parts of GitLab in the left menu. The figure depicts the current project, "Fasten Screw Dashboard", and shows that there are 69 issues (either change requests or review tasks) and one merge request open. To implement the request, a CPPS stakeholder can create a branch in the left menu under *Code* item or directly in the change request detail page by clicking on the dropdown arrow of the blue "Create merge request" button. Once the implementation on the branch is finalized, the engineer who implemented the change creates a merge request to integrate the changes into the common model.

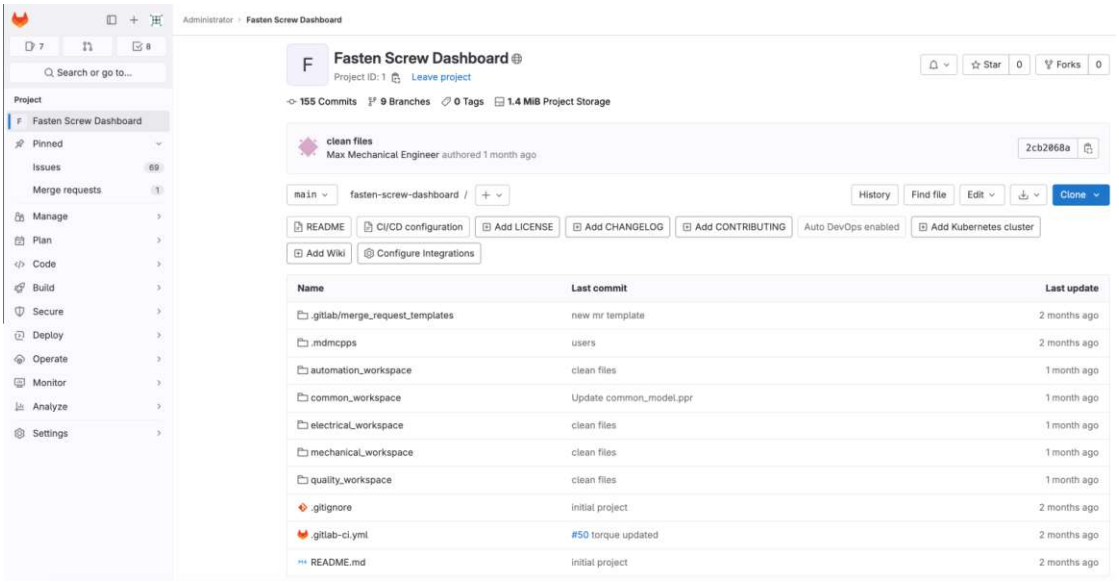

Figure 6.18: An exemplary project repository in GitLab.

Merge Request and Pipeline. Figure 6.19 depicts an exemplary merge request. A merge request consists of a title and a body, which is created using a template. The proposed template body includes the description of the merge request, which contains a link to the change request. Additionally, it describes the review process via a checklist, which has to be managed by the merge request reviewer. The reviewer is the person who created the change request. The merge request also has a tab under its title, which shows commits, pipeline runs on the merge request, and the change tab that includes the whole text-based change set compared to the main branch.

Domain-to-Common Model Comparison. To integrate a change to the common model using a *merge request*, the M-CIA method first identifies the semantic differences between the domain-specific model and the common model using the *Model Comparer* component. The resulting *diff model* is stored as *json* object in a diff.json in the domain workspace.

The *diff model* consists of a list of *diff objects*. A *diff object* is, for instance, an *attribute change* indicated by the `changeType` field and the related attribute with the *attribute* field. We indicate the parent element with the `pprAsset` field. The *valueOld* and *valueNew* fields contain the new and old values of the attribute.

Listing 6 depicts the list consisting of one *diff objects*. The *pprAsset FastenScrewDashboard_Model .electric_screwdriver* is the connecting reference between the domain model and the common model. The *oldValue* and *newValue* represent the value change from *1500.0* to *1000.0* for the `attribute` *ElectricalConcepts.power_consumption*.

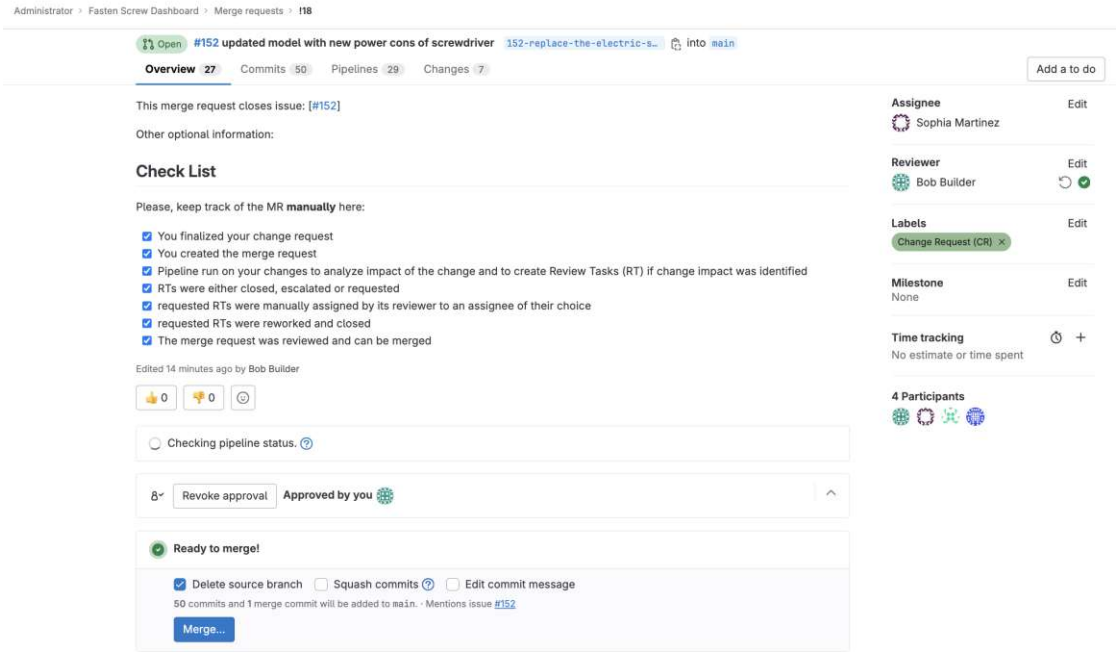

Figure 6.19: An exemplary merge request.

```
{"diffs": [{
"pprAsset": "FastenScrewDashboard_Model.electric_screwdriver",
"valueOld": 1500.0,
"changeType": "CHANGE",
"attribute": "MechanicalConcepts.power_consumption",
"valueNew": 1000.0
}]
}
```

Listing 6: *Diff model* file with an attribute value change of *power consumption* in the electrical domain.

Knowledge Graph Generation. To facilitate the semantic analysis of the change impact on models in other domains, the common model, CCG and CGs are mapped to a MDEG using the *MDEG Model Mapper* and instantiated in a graph database using Neo4j, depicted in Figure 6.20.

The colors of the knowledge graph represent the type of the nodes: resources *Electric Screwdriver* (yellow) and *Bit* (yellow), their common concepts (purple), the related mechanical concepts (red) with the attributes torque (left-blue) and bit type (right-blue). We colored the node containing the value of an attribute instance with brown.

Figure 6.20 also shows the *Concept Semantic Link CCG_DEPENDS_ON* edge between torque and bit type. These blue attribute nodes are defined in the Concept Glossary of the mechanical domain, and the glossary contains a relation between these two

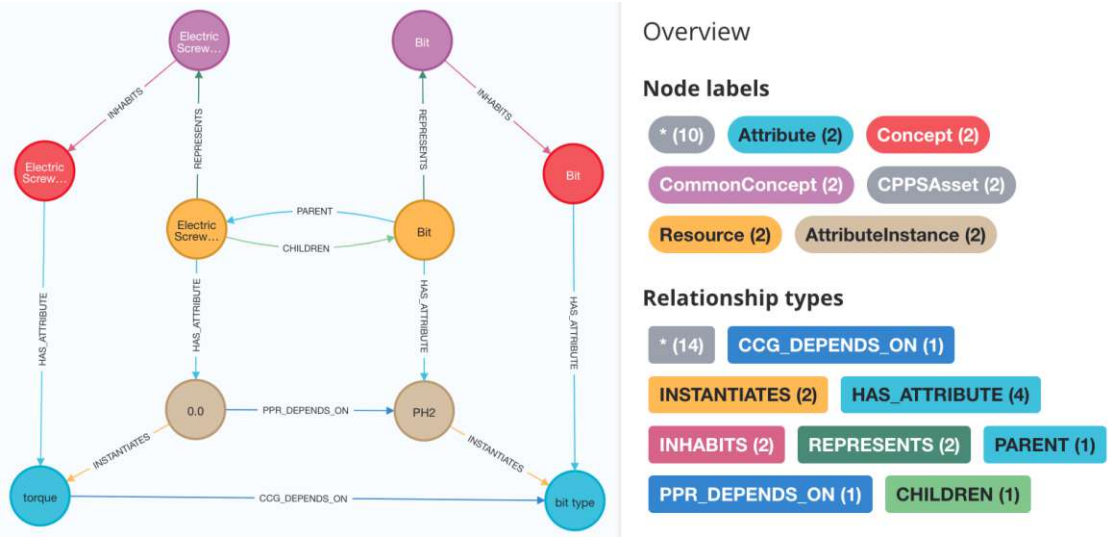

Figure 6.20: Multi-Domain Engineering Graph in Neo4j showing the exemplary resource *Electric Screwdriver* and its child resource *Bit*, including the relations on concept (CCG_DEPENDS_ON) and asset (PPR_DEPENDS_ON) level.

attributes. To transform this edge to the asset level as a *Common Model Semantic Link*, we create a new edge between the brown value instance of the attribute nodes and name it *PPR_DEPENDS_ON*.

Change Propagation and Task Creation. Finally, the dependencies are identified and based on the dependency link type; either the value change is propagated to other domains, or a review task is automatically created and assigned to an engineer with a capacity. Additionally, if a change is propagated to an attribute that has a review dependency on another attribute, a review task is created for the secondarily impacted attribute.

Figure 6.21: Activity log after the merge request pipeline run successfully.

Figure 6.21 depicts the result of the merge request pipeline. We see that the changed power consumption was propagated to the attribute current supply and screw curve. Additionally, two review tasks were created, *Review Task* 172 and 173. The author of the activity log entry is the *CIA Bot*. Alternatively, if neither a review task creation nor change propagation was performed, the bot comments with a corresponding note that the changes have no impact on other domains.

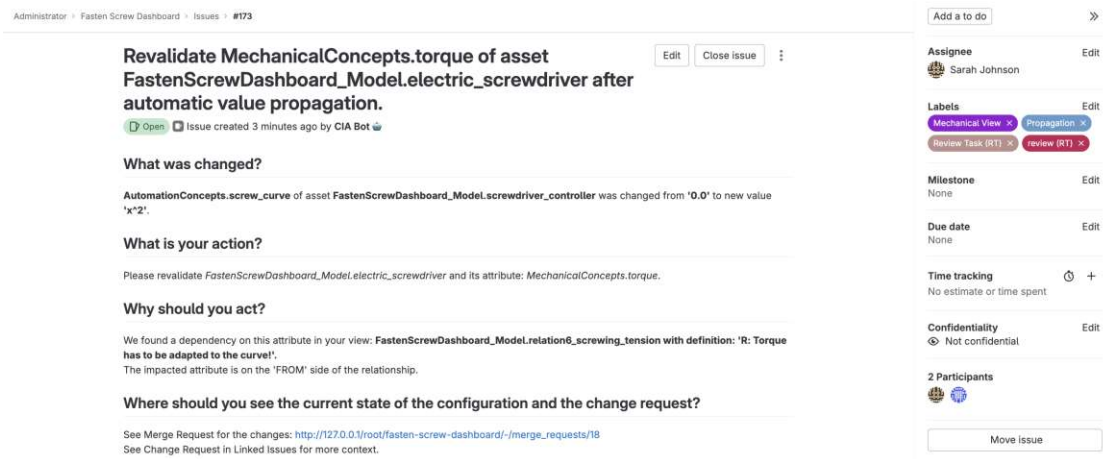

Figure 6.22: An exemplary review task created and assigned to an impacted stakeholder by the *CIA Bot*.

An exemplary review task is depicted in Figure 6.22. The task contains a title and a descriptive body to provide the reviewer with the context of the change. Additionally, the review task label and domain label were added to document what domain is impacted. We see that the author of the review task was *CIA Bot*. Sarah Johnson, a mechanical

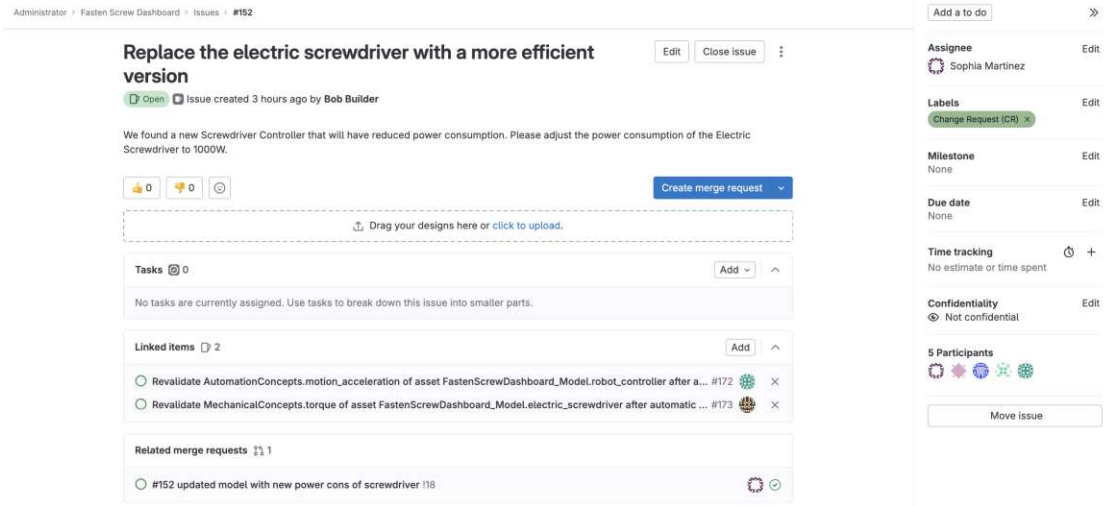

Figure 6.23: Review tasks linked to the change request by the *CIA Bot*.

engineer, was automatically assigned to the review task by *CIA Bot*. Finally, the link between the review task and the change request is created (c.f. Figure 6.23). This information is important for duplicate elimination. If a review task with the same content is already linked to the change request, we do not create the additional task.

If Sarah decides to either close the review task because the implementation is acceptable, or she decides to rework the implementation, the *CIA Bot* observes the updates to the review task and centralizes it in the merge request activity (c.f. Figure 6.24).

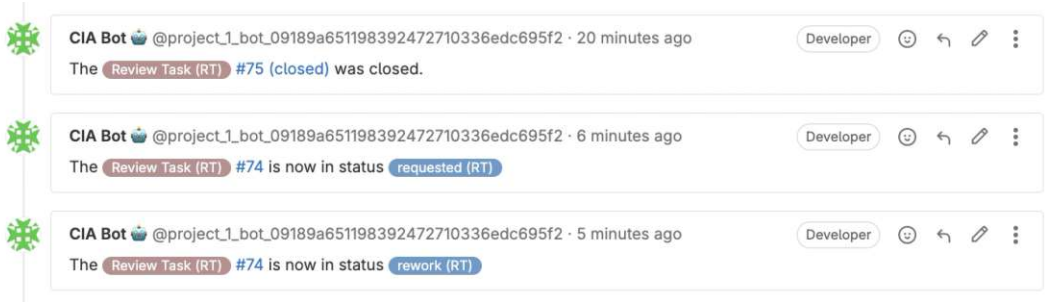

Figure 6.24: Status change tracking of the review tasks by the *CIA Bot*.

On the other hand, the automatic change propagation can be reviewed by clicking on the commit hash of the commit message "#152 Propagation: power_consumption -> power_supply" (c.f. Figure 6.21). After clicking on the commit hash, the propagation change set is shown in detail.

Figure 6.25 shows the title of the commit message, the body, and the author, and that the change propagation was performed in the *electrical view* file, which represents the electrical domain-specific model. The body of the commit message explains that the propagation was defined by a propagation link with the definition that represents the division of power_consumption (in Watt) by 240 (Volt), which is 1000 W/240 V and results in 4.17 Ampere for the new current supply value.

Issue Labels. Both change request and review tasks have labels assigned (c.f. Figure 6.22, 6.23).

To differentiate between the issue type, the M-CIA proposes the labels *Change Request (CR)* and *Review Task (RT)*. To differentiate between the status of the change request, the M-CIA method proposes labels *WiP (CR)* for requests that are currently being worked on, and *Review (CR)* for requests for which there currently exists an open merge request. If the request has no status task, this indicates that the task is open.

To differentiate between the status of the review task, the M-CIA method proposes labels *Review (RT)*, for tasks that are currently being reviewed, *requested (RT)* for tasks based on which the change implementation needs a rework so the rework is requested, *Rework (RT)*, for review tasks that were reviewed, requested, and now an engineer works on the

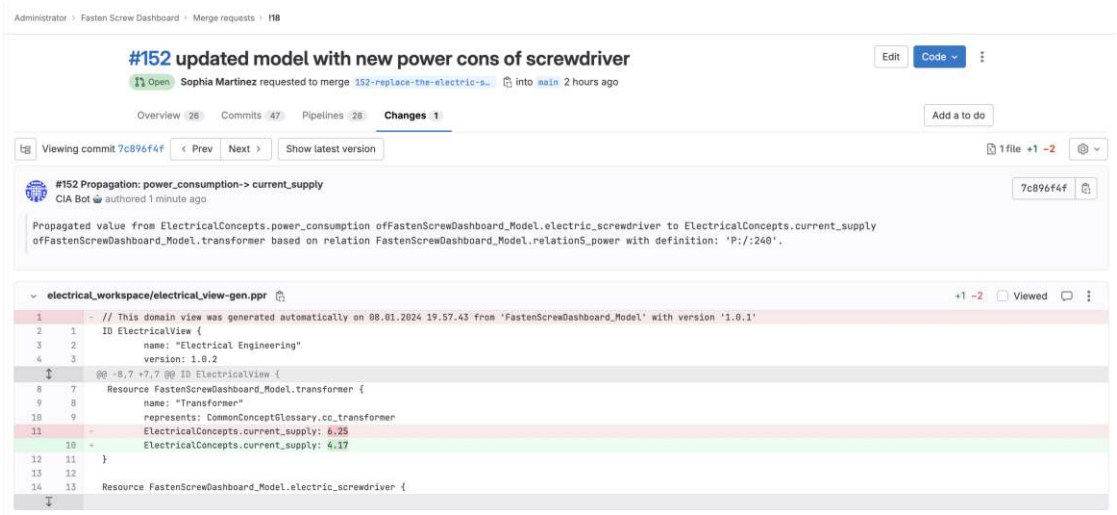

Figure 6.25: An exemplary change propagation performed by the *CIA Bot*.

task to adjust the implementation, and finally, *Escalate (RT)*, if the review task cannot be closed, additional input, discussion or decision by a *decision maker* is necessary.

Merge Request Closure. Once the merge request is reviewed and all the review tasks are closed, the merge request reviewer can check all items off the merge request checklist (c.f. Figure 6.19). Finally, the merge request is approved by the reviewer and can be merged via the blue "Merge" button.

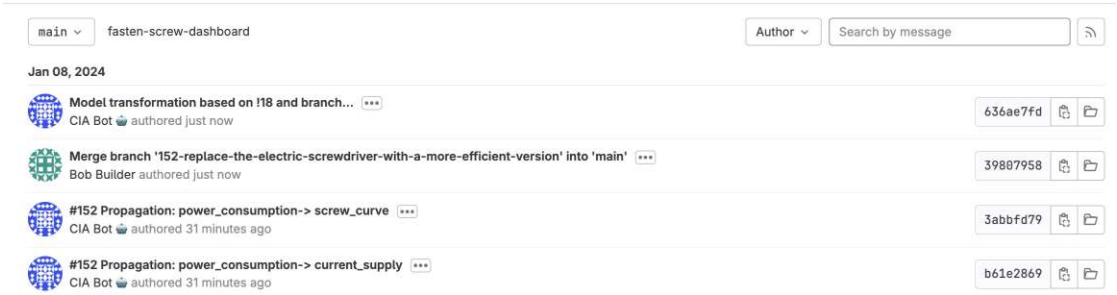

Figure 6.26: Commit log depicting the model-based integration after the text-based merge was performed by the *CIA Bot*.

During the merge activity, first, the text-based Git merge is performed. Then, the *CIA Bot* performs the model-based merge, as shown in Figure 6.26 depicts the commit log on the main branch. First, the commits made on the change requests are transferred to the main branch (e.g., propagation commits), then the text-based merge commit is performed, and finally, the *CIA Bot* merges the models and comments the action in a commit message. The merge-based merge is depicted in Figure 6.27. In the model header, the version of the model is automatically increased.

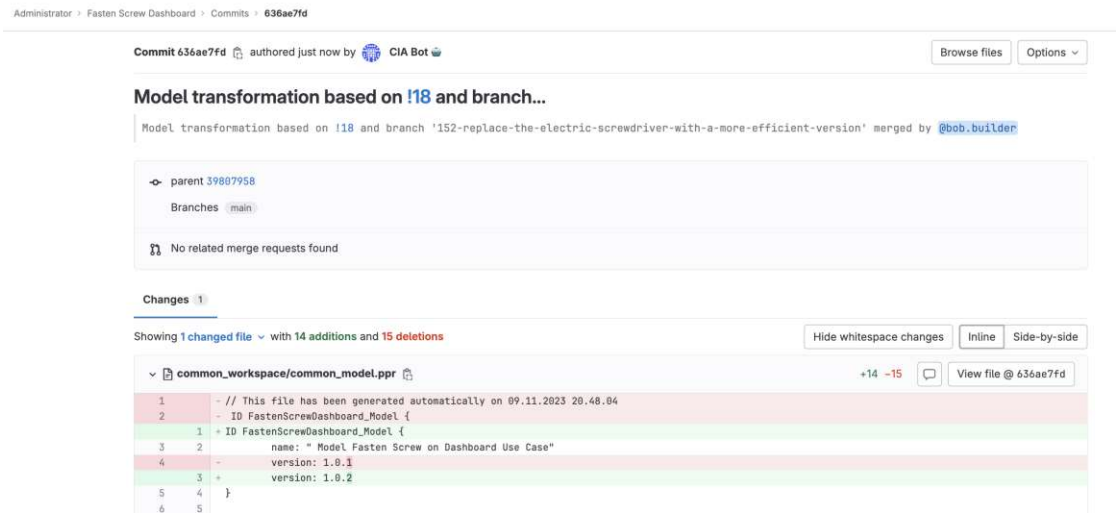

Figure 6.27: Model-based merge including the automatic version incremented by the *CIA Bot*.

Additionally, the common model inherits the attribute values from all domain-specific models, as depicted in Figure 6.28. The figure shows that the *transformer* in the common model now contains the new current supply value in Ampere.

Figure 6.28: Model-based merge with the inheritance of values into the common model from the domain-specific model (electrical) performed by the *CIA Bot*.

CHAPTER 7

Evaluation

This chapter evaluates the solution approach presented in Section 6.2 and Section 6.3 with a case study as a complement to the feasibility evaluation of the solution approach in Section 6.4.

We apply the solution approach to a real-world use case with an industry partner from the fertilizer production industry. To evaluate the M-CIA method and the M-CIA system design, we implemented the *M-CIA Management System* prototype, as described in Section 6.4.

This chapter describes the evaluation use case in Section 7.1, the evaluation procedures, and their results in Section 7.2 to 7.4. The evaluation will consist of qualitative and quantitative comparisons of the proposed solution approach to the traditional document-based approaches. Additionally, the applicability of the proposed approach in batch manufacturing will be discussed. We aim to enable the readers to reproduce the tests by describing the evaluation tasks in sufficient detail. The qualitative evaluation will be conducted based on the requirements from Subsection 6.2.1. Finally the chapter describes the results in detail for both evaluation procedures.

7.1 Evaluation Use Case: Fertilizer Mixing Case Study

To evaluate the M-CIA method, we will conduct a case study with a fertilizer production company. The company specializes in fertilizer production for bio-agriculture and produces more than ten solid and liquid products. The company has been producing fertilizers for around twenty years. The research and development of the product are conducted in a laboratory setting, and the approximate time to market is five to nine years due to the various tests that depend on the lifecycle of the agricultural plants.

Therefore, the production company had quite a stable production system setup that changed once in a couple of years and was operated manually. Additionally, the production

system is reconfigurable for the ten products the company produces based on the client's orders and has to maintain its reconfigurability for future products. However, with more demand, the fertilizer company needs to transform its manual production system into an automated CPPS to enable scaling and increase efficiency and flexibility.

Therefore, we partnered up and offered to describe their production assets formally and lay the foundation for future re-engineering and automation. In return, we could evaluate the proposed M-CIA method on a real-world use case from the industry and gain valuable feedback. We worked closely with the company's basic planner, who was in charge of preparing drawings and descriptions of the production process, as well as the future vision of the production plant, which they handed over to an external detail planning and engineering company (cf. Chapter 3, Figure 3.1 *Detailed engineering*).

We had no contact with the detail planning and engineering company throughout the process. Therefore, the basic planner was our main contact person. The basic planner was knowledgeable about the production process itself, as well as the mechanical and electrical engineering aspects of the production system.

To evaluate the proposed M-CIA method, we have defined domain and common concepts together with the basic planner, as well as the common model. Additionally, the basic planner documented the attribute dependencies across the relevant production assets. Then, we simulated changes that could occur in production and during the plant operation. Finally, we looked at how efficient the proposed method and system design are.

7.1.1 Rationale

The selection of the evaluation use case from the industry was limited to our network and the time frame of the thesis project, which implied the necessary availability of the contact person who would work with us on demand.

Additionally, the illustrative use case of *Position Screw & Dashboard* from Chapter 5 (cf. Figure 5.2) depicts a discrete production process, as input products for a given process are car parts, that need to be manipulated and assembled into the final product, based on quality constraints.

However, there are also continuous and batch processes in the industry, where the input products for the production processes are not system parts but rather liquids or solid chemical components. With this case study, we want to evaluate the feasibility of the method for the batch production process in addition to the discrete production process shown in the illustrative use case.

7.1.2 Objective

The study takes place in an industrial setting, which is also the primary audience for the method. The study is conducted with a basic planner with production and engineering knowledge. Basic planners are an essential part of the *project setup phase*. In this case, the basic planner is also very knowledgeable about engineering processes, which enables

him to evaluate the M-CIA method also on the *engineering and change management* phase. The overall objective of the case study is to learn about the following points:

- Does the M-CIA method fulfill the requirements from Subsection 6.2.1?
- What is the estimated time spent to conduct the M-CIA method in the *engineering and change management phase* of CPPS engineering for a small industrial product line compared to traditional approaches?
- What is the perceived improvement of the M-CIA method for the basic planner of the fertilizer production line compared to traditional approaches?

7.1.3 Fertilizer Mixing Domain Analysis

At the beginning of the study, we requested process descriptions and envisioned models of the re-engineered (automated) fertilizer production plant. These artifacts were rather informal and mostly hand drawings and sketches.

Together with the basic planner, we used these artifacts as a foundation to describe the main production process and its subprocesses based on BPMN, depicted in Figure 7.1

Figure 7.1 shows the first and most important production process, preparation, and production of the primary fertilizer. The process includes mixing the input products over several hours, as well as multiple filtering steps, to ensure the quality of the final primary fertilizer. This fertilizer can be later packaged and distributed to the clients or used as the foundation for the production of secondary products (fertilizers). The process diagram also differentiates between automatic and manual steps that the basic planner envisions. Previously, all steps were executed manually.

We summarize the production process from Figure 7.1 as follows: three input products are loaded into reactors, and these input products are mixed for several hours. After the mixing is done, the operator checks the quality of the initial mixture, and if necessary, they add a chemical component C_2 to achieve the necessary characteristics of the mixture. After this quality check, the mixture is pumped to the first filter for initial pre-filtering.

The pre-filtered mixture is then pumped into a container, from which another pump transfers the material to the last filtering round. From there, the final primary product is pumped into the storage, where it either waits for packaging or secondary production. Hence, reactors, filters, pumps, vents, and containers are the crucial resources used in the process. Initially, the vents and pumps were manually handled by an operator. However, this should change after the re-engineering project of the industry partner.

Based on this knowledge, the basic planner assisted in the creation of the corresponding MDEG. The MDEG is displayed in Figure 7.2 and shows four domains: *mechanical*, *electrical*, *chemical*, and *operator*. The automation engineering domain is omitted, as it is not yet implemented in the CPPS.

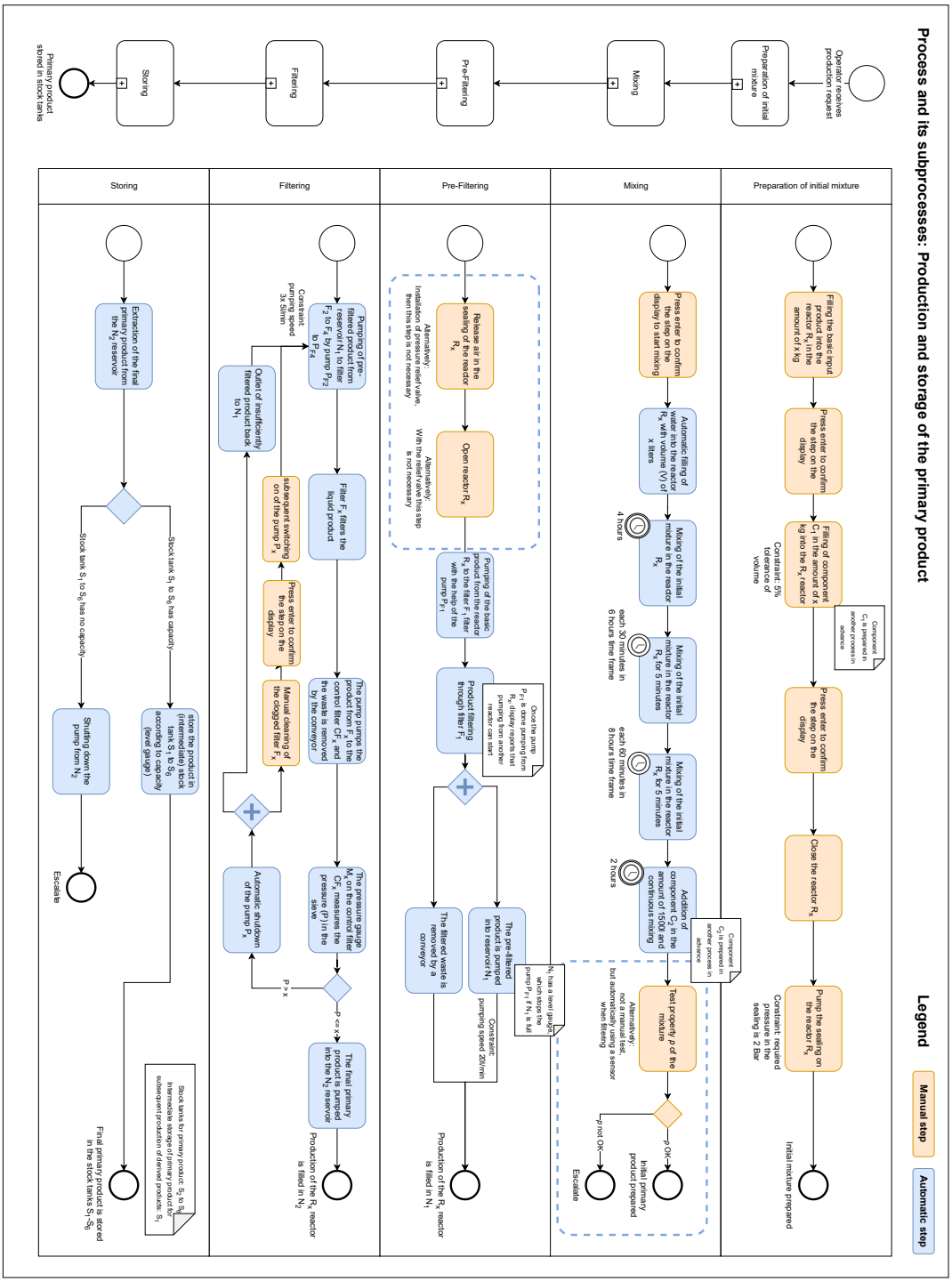

Figure 7.1: BPMN-based process description of the primary product production.

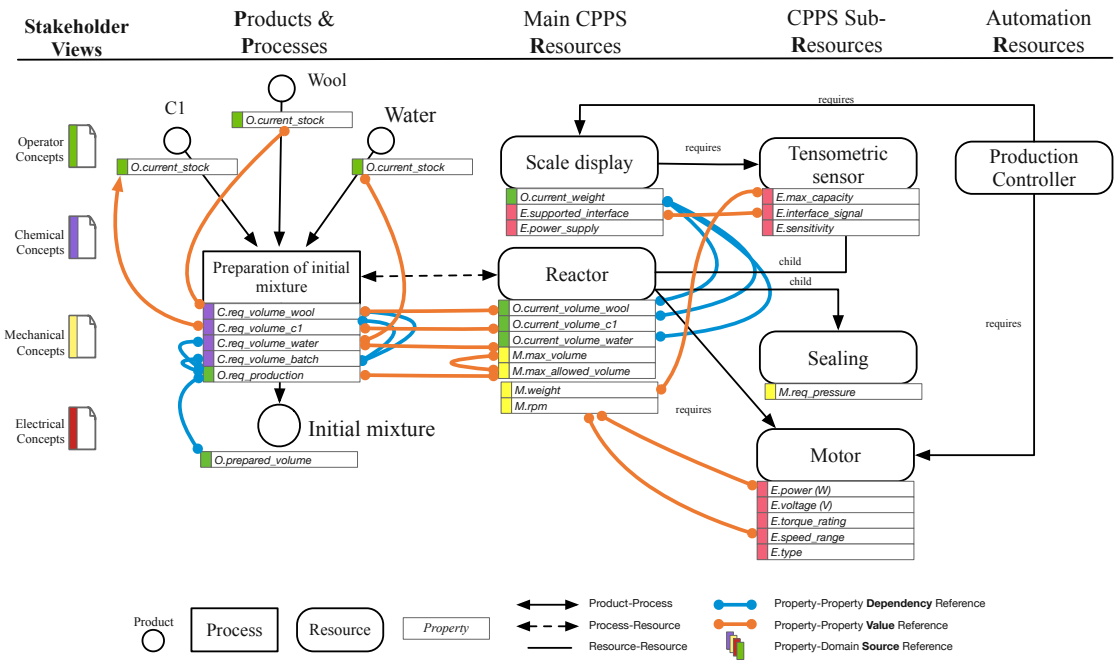

Figure 7.2: The MDEG for the subprocess *Preparation of initial mixture*.

In this use case, the *operator* also conducts the quality control. Therefore, the green color coding is retained. Additionally, the evaluation use case differs from the illustrative use case in Section 5.2 as it shows the interplay (dependencies) between the engineering domains and operators. The MDEG together with the M-CIA facilitate the impact analysis across the engineering and operation aspects of the CPPS lifecycle (c.f. Section 3.1).

7.1.4 Evaluation Questions

Reviewing the research questions from 4.1, the goal of the solution approach was to design an efficient method and system design for the *M-CIA Management* method (c.f. RQ2, RQ3). We define the *efficiency* of the method as a combination of reduced execution time and perceived improvement of the method compared to traditional approaches. We formulated the following question to evaluate the efficiency of the method and system design with the evaluation use case:

EQ1: How efficient is the M-CIA approach compared to traditional approaches in terms of execution time?

This evaluation question will be answered by comparing the proposed approach to the traditional approaches by executing tasks with the same goal and calculating the expected execution time using the Key-Stroke Level Model (KLM).

EQ2: What is the perceived improvement of the M-CIA method increased compared to the traditional approaches?

This evaluation question will be answered by the basic planner from the fertilizer production company, who will fill out a *5-point Likert scale* and evaluate the approaches based on the requirements.

EQ3: How feasible is the method for use in batch production processes?

This evaluation question will be answered by discussing the learnings from the execution of the M-CIA method on the discrete and batch production process.

7.1.5 General Evaluation Setup

As mentioned previously, we will compare the proposed method to the traditional approaches. The traditional approaches will be reconstructed based on the information from related work and use cases (c.f. Chapter 5) in collaboration with the research group members at the TU Wien and CDL-SQI.

To realize the traditional use case, we will use *Microsoft Excel* sheets as well as *Microsoft Word* documents to describe the system information. Both types of documents will have the history feature turned on to enable basic tracing capabilities in terms of document history via *Microsoft SharePoint*. One of the findings from the expert survey was that such Office tools are widely used for CIA coordination and documentation (c.f. Section 6.1).

There will be several documents, further referred to as data artifacts, that will contain information from multiple domains. These documents come from the fertilizer production company and are confidential.

To reconstruct the traditional approaches and not put them at a disadvantage, we will include the same information, such as dependency information, in the data artifacts. However, this reference will only describe assets and values in other documents or parts of the same document, which will again pose a risk of renaming and not being able to map to the correct asset or attribute, as it often happens in real situations.

The following activities are not a subject of the evaluation: 1) preparation of the data artifacts by the basic planner, 2) enrichment of the data artifacts with additional information present in the *MDM-CPPS workspace*, 2) and the design of the method and system prototype. Also, domain modeling of the system in the MDM-CPPS DSL is not included in the evaluation, as it is the subject of the related work on which this research is based Meixner et al. [2021b], and is not a primary contribution of the thesis. We will initiate the evaluation with the creation of a change request and execute the minimal engineering process as described in Chapter 5, Figure 5.3.

First, the author of the thesis will guide the basic planner through the representative evaluation tasks using all three approaches. The author of the thesis will calculate

the estimated execution time. Then, the basic planner will be asked to evaluate the approaches on a *5-point Likert scale* based on the requirements from Subsection 6.2.1.

7.2 EQ1: Estimated Execution Time of M-CIA Method

We will evaluate the M-CIA method regarding the execution time of representative tasks necessary to conduct the engineering process as introduced in Section 5 (cf. Figure 5.3). The KLM will be used as an evaluation method. KLM predicts the execution time needed to carry out a specific task scenario using a system design that is a subject of the evaluation.

To perform such an evaluation, it is necessary to list the sequence of keystroke-level actions, such as pointing a mouse to the target or pressing a key, which the user has to perform to complete a task, and then add the times required by each of the actions actions [Kieras, 2003]. These basic actions are called operators and are listed in Figure 7.1. The execution times defined in the Figure 7.1 were estimated from experimental data [Kieras, 2003].

An alternative approach to evaluating the solution approach would be to calculate the *Metrics for Usability Standards in Computing (MUSiC)* [Macleod et al., 1997]. They define two main metrics: *Effectiveness*, defined as the capability of the software system to carry out the specified task successfully, and *Efficiency*, defined as the amount of effort necessary for task completion.

Effectiveness can be measured as the number of tasks the participants successfully finished versus all tasks that participants aimed to finish, represented by a completion rate. Alternatively, error count can be used as a metric to document how many unintended actions or mistakes a user made for each task [Frøkjær et al., 2000]. *Efficiency* can be measured by calculating the task completion time necessary to carry out the task by tracing the start and end time of the task [Frøkjær et al., 2000].

However, we decided not to test these metrics of the solution approach in a setup where the basic planner would be the test user, as the basic planner assessed himself as rather technology-averse, which would give us neither representative metrics for an average user nor the best case expert user. Therefore, we selected the KLM, which is per design done by an expert user and calculates the best case execution time.

We define the following representative tasks for the evaluation: (T1) Specification of the change request, (T2) Stakeholder assignment, (T3) Change request implementation, (T4) Ready-for-review status update, (T5) Review dependencies identification, (T6) Change propagation, (T7) Review closure, (T8) Integration. We specify the context with *actor*, *pre-condition*, *success end condition*, and *main success scenario* of the tasks based on [Cockburn, 2000]. Each of the tasks will be carried out by the author of the thesis with the three approaches:

Notation	Operator	Execution Time (s)
K	Keystroke on a keyboard	0.12 - 1.2s; we use 0.28s
T(n)	Typing a character sequence of length n	n * K
P	Point with the mouse to a target on the display	1.1s
B	Press or release mouse button	0.1s
BB	Click mouse button	0.2s
H	Home hands to keyboard or mouse	0.4s
M	Mental act of routine thinking or perception	0.6 - 1.35s; we use 1.2s
W(t)	Waiting for the system to respond	we use t=0.5s
S	Scrolling	we use 3.96s

Table 7.1: List of the standard KLM operators, including their notation and estimated execution time, based on Kieras [2003]. We added the scrolling operator based on Sauro [2009].

- *Basic traditional approach (Traditional):* Usage of *Microsoft Outlook*[1] for asynchronous communication and *Microsoft SharePoint*[2] for collaboration on the artifacts.
- *Improved traditional approach (Traditional+):* Usage of *Microsoft Teams*[3] for partially synchronous communication and *Microsoft SharePoint* for collaboration on the artifacts. Additionally, we will use *Microsoft Tasks*[4] integrated to *Microsoft Teams* to manage change requests.
- *Proposed Approach (M-CIA):* Usage of the evaluation prototype that enables the execution of the M-CIA method. The most relevant actions of the method and the prototype are documented in Figure 6.13.

T1: Specification of the change request.

The goal of this task is to create a change request as a product owner that specifies what is requested to be implemented in the system. The task should contain a title and a description.

[1]Microsoft Outlook: https://www.microsoft.com/en-us/microsoft-365/outlook/email-and-calendar-software-microsoft-outlook

[2]Microsoft SharePoint: https://www.microsoft.com/en-us/microsoft-365/sharepoint/collaboration

[3]Microsoft Teams: https://www.microsoft.com/en-us/microsoft-teams/group-chat-software

[4]Microsoft Tasks: https://support.microsoft.com/en-au/office/use-the-tasks-app-in-teams-e32639f3-2e07-4b62-9a8c-fd706c12c070

Basic traditional approach (T1)

Context: Definition of a Change Request in a Microsoft Word document available in Microsoft SharePoint.

Pre-condition: The Main page of Microsoft SharePoint is open in a browser.

Success end condition: A New Word document is created in SharePoint with the name "CR1: Change of the standard production volume parameter".

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.2.

Improved traditional approach (T1)

Context: Definition of a Change Request in Microsoft Teams Task.

Pre-condition: The Main page of Microsoft Teams is open in a browser.

Success end condition: New Task is created in Teams with the name "CR1: Change of the standard production volume parameter".

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.3.

Proposed approach (T1)

Context: Definition of a Change Request in GitLab.

Pre-condition: The Main page of GitLab is open in a browser.

Success end condition: New issue is created in GitLab with the name "CR1: Change of the standard production volume parameter".

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.4.

T2: Stakeholder assignment.

The goal of this task is to have an available stakeholder assign the change requests to themselves and inform the team about it to prevent parallel work.

Basic traditional approach	Operations	Time (s)
0) Navigate to the main page of SharePoint.	-	-
1) Click on "My sites".	P, BB	1.1 + 0.2
2) Wait for the sites to load.	W(0.5)	0.5
3) Select the correct project.	M, P, BB	1.2 + 1.1 + 0.2
4) Navigate to the folder "Tasks".	M, P, BB	1.2 + 1.1 + 0.2
5) Wait for the Tasks to load.	W(0.5)	0.5
6) Click on "New".	P,BB	1.1 + 0.2
7) Select "Document".	P, BB	1.1 + 0.2
8) Wait for the new Document to load.	W(0.5)	0.5
9) Write title.	H, T(t)	0.4 + 0.28 * 55
10) Enter.	K	0.28
11) Write a description.	T(d)	0.28*100
12) Navigate to "File".	H,P, BB	0.4 + 1.1 + 0.2
13) Navigate to "Save as".	P, BB	1.1 + 0.2
14) Navigate to "Rename".	P, BB	1.1 + 0.2
15) Rename.	T(t)	0.28*55
16) Save with title.	H, P, BB	0.4 + 1.1 + 0.2
	Total:	**75.88s**

Table 7.2: Task 1 - Basic traditional approach

Improved traditional approach	Operations	Time (s)
0) Navigate to the main page of Teams.	-	-
1) Select "Teams".	P, BB	1.1 + 0.2
2) Wait for the "Teams" to load.	W(0.5)	0.5
3) Select the correct project.	M, P, BB	1.2 + 1.1 + 0.2
4) Wait for the project to load.	W(0.5)	0.5
5) Click on "Tasks" and wait.	P, BB, W(0.5)	1.1 + 0.2 + 0.5
6) Click on "Add task".	P, BB	1.1 + 0.2
7) Fill title.	H, T(t)	0.4 + 0.28*55
8) Click "Add task".	H, P, BB	0.4 + 1.1 + 0.2
9) Open the newly added task.	P, BB	1.1 + 0.2
10) Navigate to the description field.	P, BB	1.1 + 0.2
11) Fill the description in the form.	H, T(d)	0.4 + 0.28*100
12) Navigate to the labels.	H, P, BB	0.4 + 1.1 + 0.2
13) Select the correct label.	S, BB	3.96 + 0.2
14) Click on "Save".	P, BB	1.1+0.2
	Total:	**63.56s**

Table 7.3: Task 1 - Improved traditional approach

Proposed approach	Operations	Time (s)
0) Navigate to the main page of GitLab.	-	-
1) Select the correct GitLab project.	M, P, BB	1.2 + 1.1 + 0.2
2) Wait for the project to load.	W(0.5)	0.5
3) Click on "+".	P, BB	1.1 + 0.2
4) Click on "Issue".	P, BB	1.1 + 0.2
5) Wait for the issue form to load.	W(0.5)	0.5
6) Navigate to the title field.	P, BB	1.1 + 0.2
7) Fill the title in the form.	H, T(t)	0.4 + 0.28*55
8) Navigate to the description field.	H, P, BB	0.4 + 1.1 + 0.2
9) Fill the description in the form.	H, T(d)	0.4 + 0.28*100
10) Navigate to the labels.	H, P, BB	0.4 + 1.1 + 0.2
11) Select the label "Change Request (CR)".	S, BB	3.96 + 0.2
12) Click on "Save".	P, BB	1.1+0.2
	Total:	**60.46s**

Table 7.4: Task 1 - Proposed approach

Basic traditional approach (T2)

Context: Type an email informing the relevant team members that you have implemented the change request.

Pre-condition: Microsoft Outlook is open in a browser. An email distribution list of engineers, "team1@example.com," exists; for email, the change request is relevant.

Success end condition: The engineers receive an email notification about the change request assignee email the text *"Dear all, I will implement the CR1. Best, Max".*

Main success scenario: The execution steps, the KLM operators, and their exact time calculations are shown in Table 7.5.

Improved traditional approach (T2)

Context: Assign the change request to yourself, as an available engineer, in Microsoft Teams Task.

Pre-condition: Microsoft Teams is open in a browser.

Success end condition: You are assigned to the change request in Microsoft Teams Task.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.6.

Basic traditional approach	Operations	Time (s)
0) Navigate to the main page of Outlook.	-	-
1) Click on "New Mail".	P, BB	1.1 + 0.2
2) Wait for the dialog to load.	W(0.5)	0.5
3) Navigate to the "To" field.	P, BB	1.1 + 0.2
4) Fill in the email address "team1@example.com".	H, T(a)	0.4 + 0.28*17
5) Navigate to the "Subject" field.	H, P, BB	0.4 + 1.1 + 0.2
6) Fill the subject with "CR1: implementation".	M, H, T(s)	1.2 + 0.4 + 0.28*19
7) Navigate to the email body.	H, P, BB	0.4 + 1.1 + 0.2
8) Type a short email body.	H, M, T(b)	0.4 + 1.2 + 0.28*49
9) Navigate to "Send" and click.	H, P, BB	0.4 + 1.1 + 0.2
	Total:	**35.6s**

Table 7.5: Task 2 - Basic traditional approach

Improved traditional approach	Operations	Time (s)
0) Navigate to the main page of Teams.	-	-
1) Select "Teams".	P, BB	1.1 + 0.2
2) Wait for the "Teams" to load.	W(0.5)	0.5
3) Select the correct project.	M, P, BB	1.2 + 1.1 + 0.2
4) Wait for the project to load.	W(0.5)	0.5
5) Click on "Tasks" and wait.	P, BB, W(0.5)	1.1 + 0.2 + 0.5
6) Click on the new CR1.	P, BB	1.1 + 0.2
7) Navigate to the "Assign" field.	P, BB	1.1 + 0.2
8) Type your name "Max".	H, T(n)	0.4 + 0.28*3
9) Select the user from the dropdown.	H, P, BB	0.4 + 1.1 + 0.2
10) Close task.	P, BB	1.1 + 0.2
	Total:	**13.44s**

Table 7.6: Task 2 - Improved traditional approach

Proposed approach (T2)

Context: Assign the change request to yourself as an available engineer in GitLab.

Pre-condition: GitLab is open in a browser.

Success end condition: You are assigned to the change request in GitLab.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.7.

Proposed approach	Operations	Time (s)
0) Navigate to the main page of GitLab.	-	-
1) Select the correct GitLab project.	M, P, BB	1.2 + 1.1 + 0.2
2) Wait for the project to load.	W(0.5)	0.5
3) Click on "Plan".	P, BB	1.1 + 0.2
4) Click on "Boards".	P, BB	1.1 + 0.2
5) Wait for the boards to load.	W(0.5)	0.5
6) Click on the CR1.	P, BB	1.1 + 0.2
7) Wait for the CR1 to load.	W(0.5)	0.5
8) Navigate to assign to yourself and click.	P, BB	1.1 + 0.2
	Total:	**9.2s**

Table 7.7: Task 2 - Proposed approach

T3: Change request implementation.

The goal of the task is to implement a change as an assigned stakeholder, as specified in a change request. For this task, we do not differentiate between basic and improved traditional approaches in how the task is carried out. We use the same steps and the same operators.

Basic and improved traditional approach (T3)

Context: Read the task description and change a value in the correct document in Microsoft Sharepoint.

Pre-condition: Open the main page of Sharepoint in a browser. In another tab, the change request is opened, either as a Microsoft Word document (basic traditional approach) or as a Microsoft Teams Task (improved traditional approach).

Success end condition: Correct data artifact is updated and closed.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.8.

Proposed approach (T3)

Context: Read the task description and change the value of an asset's property in the discipline-specific model file located in the *MDM-CPPS workspace*.

Pre-condition: Open the change request in GitLab.

Basic and improved traditional approach	Operations	Time (s)
0) Navigate to the main page of SharePoint, read CR.	-	-
1) Click on "My sites".	P, BB	1.1 + 0.2
2) Wait for the sites to load.	W(0.5)	0.5
3) Select the correct project.	M, P, BB	1.2 + 1.1 + 0.2
4) Navigate to the folder "Data Artifacts".	M, P, BB	1.2 + 1.1 + 0.2
5) Wait for the folder to load.	W(0.5)	0.5
7) Select the correct data artifact.	P, BB	1.1 + 0.2
8) Wait for the data artifact to load.	W(0.5)	0.5
9) Look for the element to update.	S, M, P, BB	3.96 + 1.2 + 1.1 + 0.2
10) Change the value of the element.	H, T(v)	0.4 + 0.28*2
11) Close the file.	H, P, BB	0.4 + 1.1 + 0.2
	Total:	**18.22s**

Table 7.8: Task 3 - Basic and improved traditional approach

Success end condition: Implementation is pushed to the feature branch of the change request. The commit message contains the issue number and a short note: "#150 value changed".

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.9.

T4: Ready-for-review status update.

The owner of the implementation provides a status update to the product owner that he has implemented the change and it is now ready for review.

Basic traditional approach (T4)

Context: In this approach, the owner of the implementation sends an email to the product owner.

Pre-condition: Microsoft Outlook is open in a browser.

Success end condition: Email is sent to notify the change requester via the email distribution list (team1@example.com) that the change request is ready for a review with the text "Hi Joe, the CR1 is ready for a review. Best, Max" and subject "CR1: Review".

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.10.

Proposed approach	Operations	Time (s)
0) Open the change request in GitLab.	-	-
1) Click on the "Create merge request" drop-down.	P, BB	1.1 + 0.2
2) Select the "Create branch" option.	P, BB	1.1 + 0.2
3) Confirm "Create branch".	P, BB	1.1 + 0.2
4) Open VS Code.	P, BB	1.1 + 0.2
5) Perform Git pull.	P, BB, P, B	2 * (1.1 + 0.2)
6) Check out the new Git branch.	P, BB, P, BB	2 * (1.1 + 0.2)
7) Trigger domain-specific model generation.	P, BB, P, BB	2 * (1.1 + 0.2)
8) Select the correct domain workspace.	P, BB	1.1 + 0.2
9) Select the correct domain-specific model.	P, BB	1.1 + 0.2
10) Look for the element to update.	S, M, P, BB	3.96 + 1.2 + 1.1 + 0.2
11) Change the value of the element.	H, T(v)	0.4 + 0.28*2
12) Navigate to the commit message.	H, P, BB	0.4 + 1.1 + 0.2
13) Type the commit message.	H, T(m)	0.4 + 0.28*18
14) Commit and push changes.	H, P, BB, P, BB	0.4 + 2 * (1.1 + 0.2)
	Total:	**33.16s**

Table 7.9: Task 3 - Proposed approach

Improved traditional approach (T4)

Context: In this approach, the owner of the implementation moves the change request to the "Review" bucket in Microsoft Teams Tasks.

Pre-condition: Microsoft Teams is open in a browser. Microsoft Teams Taskboard includes the following buckets: Open, WiP, Review, and Done. The CR1 Task is in the WiP bucket.

Success end condition: The change request task is moved to the "Review" bucket in Microsoft Teams Tasks.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.11.

Proposed approach (T4)

Context: In this approach, the owner of the implementation moves the issue to the "Review (CR)" column on the CR Board and creates a merge request in GitLab.

Pre-condition: GitLab is open in a browser.

Success end condition: The change request has the label "Review (CR)", and the merge request exists. The merge request is assigned for the review to the change requester.

Basic traditional approach	Operations	Time (s)
0) Navigate to the main page of Outlook.	-	-
1) Click on "New Mail".	P, BB	1.1 + 0.2
2) Wait for the dialog to load.	W(0.5)	0.5
3) Navigate to the "To" field.	P, BB	1.1 + 0.2
4) Fill in the e-mail address "team1@example.com".	H, T(a)	0.4 + 0.28*17
5) Navigate to the "Subject" field.	H, P, BB	0.4 + 1.1 + 0.2
6) Fill the subject with "CR1: Review".	M, H, T(s)	1.2 + 0.4 + 0.28*11
7) Navigate to the body.	H, P, BB	0.4 + 1.1 + 0.2
8) Type a short email body.	H, M, T(b)	0.4 + 1.2 + 0.28*48
9) Navigate to "Send" and click.	H, P, BB	0.4 + 1.1 + 0.2
	Total:	**33.08s**

Table 7.10: Task 4 - Basic traditional approach

Improved traditional approach	Operations	Time (s)
0) Navigate to the main page of Teams.	-	-
1) Select "Teams".	P, BB	1.1 + 0.2
2) Wait for the "Teams" to load.	W(0.5)	0.5
3) Select the correct project.	M, P, BB	1.2 + 1.1 + 0.2
4) Wait for the project to load.	W(0.5)	0.5
5) Click on "Tasks" and wait.	P, BB, W(0.5)	1.1 + 0.2 + 0.5
6) Drag & Drop the CR1 to Review bucket.	P, B, P, B	1.1 + 0.1 + 1.1 + 0.1
	Total:	**9.00s**

Table 7.11: Task 4 - Improved traditional approach

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.12.

T5: Review dependencies identification.

The product owner receives the feedback from the owner of the CR implementation and reviews it in this step. The review involves scanning through the data artifacts and looking for information about explicitly documented dependencies. For those dependencies, the product owner creates new review tasks and assigns them to relevant engineers from impacted domains.

Proposed approach	Operations	Time (s)
0) Navigate to the main page of GitLab.	-	-
1) Select the correct GitLab project.	M, P, BB	1.2 + 1.1 + 0.2
2) Wait for the project to load.	W(0.5)	0.5
3) Click on "Plan".	P, BB	1.1 + 0.2
4) Click on "Boards".	P, BB	1.1 + 0.2
5) Wait for the boards to load.	W(0.5)	0.5
6) Drag the CR1 from the WiP bucket.	P, B	1.1 + 0.1
7) Drop the CR1 into the Review bucket.	P, B	1.1 + 0.1
8) Click on "+".	P, BB	1.1 + 0.2
9) Click on "New merge request".	P, BB	1.1 + 0.2
10) Navigate to "Source branch".	P, BB	1.1 + 0.2
11) Select source branch	S, BB	3.96 + 0.2
12) Click on "Compare branches and continue".	P, BB	1.1 + 0.2
13) Wait for the page to load.	W(0.5)	0.5
14) Navigate to the description template.	P, BB	1.1 + 0.2
15) Select the "CPPS Merge Request" template.	P, BB	1.1 + 0.2
16) Navigate to the description.	P, BB	1.1 + 0.2
17) Add the issue number '150' to the description.	H, T(4)	0.4 + 0.28*5
18) Assign the MR to yourself	P, BB	1.1 + 0.2
19) Navigate to the reviewer field.	P, BB	1.1 + 0.2
20) Write the reviewer's name.	H, T(3)	0.4 + 0.28*3
21) Select the reviewer.	H, P, BB	0.4 + 1.1 + 0.2
22) Click on "Create merge request".	P, BB	1.1 + 0.2
	Total:	**31.9s**

Table 7.12: Task 4 - Proposed approach

Basic traditional approach (T5)

Context: Look for dependencies in the data artifacts and write a mail requesting to review the change. For simplicity, we assume the product owner only finds one review dependency and, therefore, has to write a main to one engineer.

Pre-condition: Microsoft SharePoint and Outlook are opened in a browser.

Success end condition: Mail sent to request a review to the best guess available engineer from the impacted domain.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.13.

Basic traditional approach	Operations	Time (s)
0) Navigate to the main page of SharePoint, read CR.	-	-
1) Click on "My sites".	P, BB	1.1 + 0.2
2) Wait for the sites to load.	W(0.5)	0.5
3) Select the correct project.	M, S, P, BB	1.2 + 3.96 + 1.1 + 0.2
4) Navigate to the folder "Data Artifacts".	M, P, BB	1.2 + 1.1 + 0.2
5) Wait for the folder to load.	W(0.5)	0.5
7) Select the correct data artifact.	P, BB	1.1 + 0.2
8) Wait for the data artifact to load.	W(0.5)	0.5
9) Review the data artifact.	-	-
10) Write an email to request a review.	See Table 7.5	35.6
	Total:	**48.65s**

Table 7.13: Task 5 - Basic traditional approach

Improved traditional approach (T5)

Context: Look for dependencies in the data artifacts and create review tasks to review the dependencies. For simplicity, we assume the product owner only finds one review dependency and has to create only one review task.

Pre-condition: Microsoft SharePoint and Microsoft Teams are opened in a browser.

Success end condition: Review task created and assigned to the best guess available engineer from the impacted domain.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.14.

Proposed approach (T5)

Context: Identification of the review dependencies is executed automatically after the merge request is created.

Pre-condition: A merge request was created.

Success end condition: A Review task was created, assigned to an engineer from the impacted domain, linked to the initial change request, and linked to the merge request.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.15.

Improved traditional approach	Operations	Time (s)
0) Navigate to the main page of SharePoint, read CR.	-	-
1) Click on "My sites".	P, BB	1.1 + 0.2
2) Wait for the sites to load.	W(0.5)	0.5
3) Select the correct project.	M, P, BB	1.2 + 1.1 + 0.2
4) Navigate to the folder "Data Artifacts".	M, P, BB	1.2 + 1.1 + 0.2
5) Wait for the folder to load.	W(0.5)	0.5
7) Select the correct data artifact.	P, BB	1.1 + 0.2
8) Wait for the data artifact to load.	W(0.5)	0.5
9) Review the data artifact.	-	-
10) Create a review task.	See Table 7.3	63.56
	Total:	**72.65s**

Table 7.14: Task 5 - Improved traditional approach

Proposed approach	Operations	Time (s)
0) Create a merge request.	-	-
1) Wait for the MR pipeline to run.	W(0.5)	0.5
	Total:	**0.5s**

Table 7.15: Task 5 - Proposed approach

T6: Change propagation.

An engineer from the impacted domain receives the information from the product owner that a change was implemented that has to be propagated to the assets in their domain. Based on the change description, the engineer from the impacted domain revisits a related data artifact to propagate the updated value.

Basic traditional approach (T6)

Context: Manual propagation including manual transformation of data.

Pre-condition: Dependency information included in the information from the product owner, including the value transformation information.

Success end condition: Data artifact was updated with the new propagated value.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.16.

Basic traditional approach	Operations	Time (s)
0) Navigate to the main page of SharePoint, read CR.	-	-
1) Click on "My sites".	P, BB	1.1 + 0.2
2) Wait for the sites to load.	W(0.5)	0.5
3) Select the correct project.	M, P, BB	1.2 + 1.1 + 0.2
4) Navigate to the folder "Data Artifacts".	M, P, BB	1.2 + 1.1 + 0.2
5) Wait for the folder to load.	W(0.5)	0.5
7) Select the correct data artifact.	P, BB	1.1 + 0.2
8) Wait for the data artifact to load.	W(0.5)	0.5
9) Review the data artifact.	-	-
10) Calculate the value manually.	-	-
11) Navigate to the value to update.	P, BB, H	1.1 + 0.2 + 0.4
12) Insert the new value.	T(2)	0.28*2
	Total:	**11.35s**

Table 7.16: Task 6 - Basic traditional approach

Improved traditional approach (T6)

Context: Automatic propagation of the values in the Microsoft Excel file.

Pre-condition: The relevant data artifact, a Microsoft Excel file, contains the formula to transform value from one cell to another.

Success end condition: Data artifact was updated with the new propagated value.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.17.

Proposed approach (T6)

Context: Identification of the propagation dependencies and propagation execution is done automatically after the merge request is created.

Pre-condition: A merge request was created.

Success end condition: The change propagation was executed across the system model files.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.18.

Improved traditional approach	Operations	Time (s)
0) Change of the value in MS Excel as part of T3.	-	-
1) Automatic formula calculation in MS Excel.	-	0.0
	Total:	**0.0s**

Table 7.17: Task 6 - Improved traditional approach

Proposed approach	Operations	Time (s)
0) Create a merge request.	-	-
1) Wait for the MR pipeline to run.	W(0.5)	0.5
	Total:	**0.5s**

Table 7.18: Task 6 - Proposed approach

T7: Review closure.

A dependency reviewer informs the product owner that they finished the review.

Basic traditional approach (T7)

Context: The reviewer responds to the email received in Task 5.

Pre-condition: Microsoft Outlook is opened in a browser.

Success end condition: The product owner received an email stating that the review was done.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.19.

Improved traditional approach (T7)

Context: The reviewer moves the review task created in Task 5 to the 'Done' bucket.

Pre-condition: Microsoft Outlook is opened in a browser.

Success end condition: The review task is moved to the 'Done' bucket.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.20.

Basic traditional approach	Operations	Time (s)
1) Write an email.	See Table 7.5	35.6
	Total:	**35.6s**

Table 7.19: Task 7 - Basic traditional approach

Improved traditional approach	Operations	Time (s)
1) Move the review task to the 'Done' bucket.	See Table 7.11	9.00s
	Total:	**0.0s**

Table 7.20: Task 7 - Improved traditional approach

Proposed approach	Operations	Time (s)
0) Navigate to the main page of GitLab.	-	-
1) Select "My issues".	P, BB	1.1 + 0.2
2) Wait for the issues to load.	W(0.5)	0.5
3) Select the review task.	P, BB	1.1 + 0.2
4) Wait for the task to load.	W(0.5)	0.5
5) Click on "Close issue".	P, BB	1.1 + 0.2
	Total:	**4.9s**

Table 7.21: Task 7 - Proposed approach

Proposed approach (T7)

Context: The reviewer closes the review task created in Task 5.

Pre-condition: GitLab main page is opened in a browser.

Success end condition: The review task is closed.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.21.

T8: Final integration.

The product owner reviews the status of the review tasks and concludes that everything has been done. They confirm the new version of the artifacts and increase the version number of the impacted data artifacts.

Basic and improved traditional approach	Operations	Time (s)
0) Navigate to the main page of SharePoint.	-	-
1) Click on "My sites".	P, BB	1.1 + 0.2
2) Wait for the sites to load.	W(0.5)	0.5
3) Select the correct project.	M, P, BB	1.2 + 1.1 + 0.2
4) Navigate to the folder "Data Artifacts".	M, P, BB	1.2 + 1.1 + 0.2
5) Wait for the folder to load.	W(0.5)	0.5
7) Click on "..." on the correct data artifact.	P, BB	1.1 + 0.2
8) Click on rename.	P, BB	1.1 + 0.2
12) Change version from 1.0.0. to 1.0.1.	H, T(2)	0.28*2
13) Confirm "Rename".	H, P, BB	0.4 + 1.1 + 0.2
	Total:	**25.11s**

Table 7.22: Task 8 - Basic and improved traditional approach

Basic and improved traditional approach (T8)

Context: The product owner increases the version of the relevant data artifact by renaming it.

Pre-condition: Microsoft SharePoint main page is opened in a browser.

Success end condition: The data artifact is named "data artifact system model V1.0.1".

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.22.

Proposed approach (T8)

Context: The product owner merges the change request branch to the main branch. The version of the common model will be automatically increased.

Pre-condition: GitLab main page is opened in a browser.

Success end condition: The main branch now contains the new version of the common model and its version is incremented.

Main success scenario: The execution steps, the KLM operators, and their execution time calculations are shown in Table 7.23.

Proposed approach	Operations	Time (s)
0) Navigate to the main page of GitLab.	-	-
1) Select "Merge Requests".	P, BB	1.1 + 0.2
2) Select the "Review requests".	P, BB	1.1 + 0.2
3) Wait for the merge requests to load.	W(0.5)	0.5
4) Select the merge request.	P, BB	1.1 + 0.2
5) Wait for the merge request to load.	W(0.5)	0.5
6) Click on "Merge".	P, BB	1.1 + 0.2
	Total:	**6.2s**

Table 7.23: Task 8 - Proposed approach

7.2.1 Execution Time Results

As shown in Table 7.24, the M-CIA approach takes less time to execute on the majority of the tasks. In Task 4 and Task 6, the proposed approach is better than the basic traditional approach but worse than the improved traditional approach in regards to the estimated execution time. In Task 3, the proposed method has the longest execution time out of all three approaches.

In Task 1, it seems to be very easy and fast to create a change request issue in GitLab via the proposed approach, while creating a new document with the change request description takes the longest due to navigation to the right project and folder. The improved traditional approach performs moderately; creating a task in Microsoft Teams takes slightly longer due to the hierarchical Teams structure.

In Task 2, GitLab provides the functionality to assign a change request issue to the currently logged-in user per one click, which makes it very efficient. Assigning the user to the task in Microsoft Teams via an improved traditional approach requires more clicks, as the current user has to be found in the dropdown. The least efficient was the basic traditional approach that required asynchronous communication via e-mail.

The proposed approach performed the worst on Task 3, in which the task execution took around 15 seconds longer. This is due to three extra actions. First, we included the time necessary to create a Git branch, which is only necessary to create once for each change request. Creation of the branch took around 4.4 seconds. Second, the implementation was made in Visual Studio Code and the new branch had to be pulled and the domain-specific model generated. Third, extra time was calculated for typing the commit message with the change request ID and a short note, which allows for change traceability. If we want to add the same level of information in the basic traditional approach, a definition of a note or additional change log would have to be created. In the case of the improved

Task description	Traditional	Traditional+	M-CIA
Task 1: Specification of the CR.	75.88s	63.56s	60.46s
Task 2: Stakeholder assignment.	35.60s	13.44s	09.20s
Task 3: CR implementation.	18.22s	18.22s	33.16s
Task 4: Ready-for-review status update.	33.08s	09.00s	31.90s
Task 5: Review dependency identification.	48.65s	72.65s	00.50s
Task 6: Change propagation.	11.35s	00.00s	00.50s
Task 7: Review closure.	35.60s	09.00s	04.90s
Task 8: Final integration.	25.11s	25.11s	06.20s

Table 7.24: Results of KLM evaluation in seconds (s) for each approach and each task. Color coding: red (longest execution time), yellow (middle execution time), and green (shortest execution time) results.

traditional approach, we could add the comment to the Microsoft Teams Task to link the change to the task. In both cases, the execution time would be at least as long as in the proposed approach.

In Task 4, the proposed approach performed moderately. First, the same actions as in the improved traditional approach were performed (moving the ticket to the right status bucket), and then an additional step for creating the merge request was performed. The creation of a merge request in this step allows for minimal execution time in Task 5 and Task 6. The basic traditional approach performed the worst, as an e-mail had to be written to the right stakeholders.

In Task 5, we see the improvement in the execution time of the proposed approach for two reasons: compared to the basic and improved traditional approach, the dependencies are identified automatically. Additionally, in the basic traditional approach, an e-mail has to be sent manually to request a review in an impacted domain to the best-guess available engineer. In the improved traditional approach, a task in Microsoft Teams has to be created manually and assigned to the best-guess available engineer. The engineer assignment happens automatically in the proposed approach, and the review tasks are also created automatically in GitLab.

This is possible due to the merge request pipeline that was triggered after the merge request creation in Task 4. It takes some time for the pipeline to finish and to create the review tasks. Therefore, we calculated the standard waiting time. In reality, this is dependent on the MDEG size and parameters of the machine on which the prototype runs. Nevertheless, no manual action is required.

The same applies to the proposed approach in Task 6. The propagation dependencies are identified automatically, and the value transformations are also conducted in the course of the merge request pipeline run. Here, we also calculated the standard waiting time to wait for the value propagation. The value propagation is executed as part of the merge request pipeline, and the actual run time depends on the MDEG size and parameters of the machine on which the prototype runs. Again, no manual action is required, only

the creation of the merge request in Task 4. For the improved traditional approach, we used a Microsoft Excel sheet with pre-implemented formulas that propagate the values immediately once a property is updated. Therefore, we expect an execution time of zero. In the basic traditional approach, the value propagation to the right Microsoft Excel cell is done manually.

In Task 7, it took the least time to close the review tasks to inform the related stakeholders that the review was concluded via the proposed approach. The improved traditional approach also delivered promising results. The basic traditional approach took the longest, as again, an e-mail has to be sent to the relevant stakeholders.

In Task 8, we did not differentiate between the two traditional approaches. In both cases, the updated data artifacts were renamed to acknowledge the new version of the document. In comparison, the proposed approach took much less time, as the merge request has to be merged via one button click.

7.3 EQ2: Perceived Improvement of M-CIA Method

To complement the previously presented quantitative evaluation of the solution approach, we also evaluate the solution approach qualitatively to elicit the perceived improvement of the method. User satisfaction is an important usability metric in addition to efficiency and effectiveness [Macleod et al., 1997].

A well-known approach to user satisfaction testing is Software Usability Measurement Inventory (SUMI) [5]. This approach consists of 50 usability-related questions that a participant has to answer with answers such as agree, don't know, or disagree. Furthermore, the participants' opinions can also be collected with a Likert scale [Sullivan and Artino, 2013].

We decided to select the *5-point Likert scale* evaluation to elicit the opinion of the basic planner, as questions in SUMI are too generic and do not cover questions regarding our requirements. Additionally, the responses in SUMI are formulated as binary agree/disagree options complemented with *don't know* option, which does not allow for detailed comparison to the traditional approaches.

Table 7.25 depicts the perceived improvement of the M-CIA method compared to the basic traditional and improved traditional approach, filled in by a basic planner from the evaluation use case. The basic planner also provided the reasoning behind the assessment, which we describe below.

We have evaluated each of the six phases (P1-P6) from the illustrative minimal engineering and change management process defined in Chapter 5.1, in Figure 5.3. The evaluation criteria (a-e) for the evaluation of each phase were reflected by the basic planner under consideration of the requirements from Subsection 6.2.1.

[5] SUMI: `https://sumi.uxp.ie/about/whatis.html`

Process phases (Fig. 5.3)	Traditional	Traditional+	M-CIA
P-1 Change request preparation			
(a) Change request definition	○	○	○
(b) Change request traceability	○	+	++
P-2 Change implementation			
(a) Separation of concerns	−	−	+
(b) Intuitiveness of the approach	+	+	− −
P-3 Change impact analysis			
(a) Stakeholder identification	−	−	+
(b) Definition of dependencies	−	−	○
(c) Analysis of review dependencies	−	○	+
(d) Analysis of prop. dependencies	−	○	+
(e) Propagation of change	− −	+	++
P-4 Multi-disciplinary review			
(a) Review coordination	−	○	+
P-5 Multi-disciplinary rework			
(a) Traceability of the rework	−	○	+
P-6 Change integration			
(a) Version management	○	○	+
(b) Semantic change integration	− −	− −	++
(c) Syntax change integration	○	○	○
(d) Traceability of the implementation	−	−	+
(e) Traceability of artifact change	○	○	++

Table 7.25: Perceived improvement of the multi-domain CIA using traditional approaches compared to M-CIA method based on a 5-point Likert scale (++, +, ○, −, − −), where ++ indicates very strong improvement, and − − very negative effects.

Change request preparation phase was evaluated on the easiness of defining a change request in a corresponding tool (c.f. R3, Subsection 6.2.1). In all three approaches, the basic planner concluded that the effort is equal. However, the change request traceability was perceived as strongly improved, given the activity logs in GitLab, the possibility to see links to change sets in the model for each of the change requests, as well as the status of the change request being transparent to the whole team (c.f. R2, Subsection 6.2.1). The change request status transparency is possible in the improved traditional approach. However, links to change sets in data artifacts are missing. In the basic traditional approach, these advantages of the ticketing system are not available.

Change implementation phase was evaluated on the separation of concerns criterion (c.f. R1, Subsection 6.2.1), which was regarded as positive, given the domain-specific views and the dedicated common view, compared to heterogenous artifacts or documents that contain cross-domain information. However, it was initially challenging to decide what information belongs to what domain.

In traditional approaches, information was scattered over multiple data artifacts, or a single data artifact contained information relevant to multiple domains. The intuitiveness of the proposed approach was regarded as very negative (c.f. R3, Subsection 6.2.1). The basic planner was not familiar with Git or Git Workflow before, which was the main trigger for the negative ranking. This should be considered, and specialized training for stakeholders should be planned if launching the M-CIA method to an enterprise.

CIA phase was evaluated mostly positively. Stakeholder identification by the *CIA Bot* for the change review did not receive the best ranking, as the current capacity of the reviewer might not be the only relevant factor. Still, it was a good improvement compared to the other two traditional approaches, in which the assignment of the engineers is done by a product owner (c.f. R5, Subsection 6.2.1). A possible improvement would be the integration of the *CIA Bot* with the calendar or a system in which absences are recorded to avoid a person who is currently absent (e.g., sick leave, vacation) getting assigned to the review task.

The definition of dependencies was considered average, as the definition still has to be done explicitly (c.f. R7, Subsection 6.2.1). However, the CIA method includes a formal language for defining the semantics of the dependencies, which was not the case in the basic traditional approach. In the improved traditional approach, the definition of formulas to value transformation was used. This fulfilled the task. However, this is possible only when using a sheet format, e.g., Microsoft Excel, and when the sheet is well-managed to avoid mistakes or unwanted changes.

Analysis of the dependencies was regarded as positive. However, there is still room for improvement, as the *CIA Bot* only identifies the dependencies that were previously added to the model (c.f. R5). An improvement would be the identification of dependencies based on previous knowledge of a machine-learning model. However, the approach would require the data to learn from. The proposed approach provides the means to gather and validate such data for further improvement of the approach.

Propagation of the changes was perceived as a very good improvement, as no manual action or calculations were necessary because the *CIA Bot* performed the propagation based on the specified propagation rules (c.f. R5, Subsection 6.2.1). However, the improved traditional approach did not take any execution time; while the proposed approach required waiting for the merge request pipeline to pass, the traceability of the propagation was perceived much more positively by the basic planner. This was due to commit messages linked to a merge request authored by the *CIA Bot*. In the traditional approaches, the value propagation did not take any time, but it was cumbersome to understand what exactly was changed as part of propagation.

Multi-disciplinary review phase was evaluated positively, compared to both traditional approaches. The review coordination using our proposed method was easy thanks to the *CIA Bot* that provided the context of the change request, the reasoning behind the review request, an overview of the changes made to the model based on the change request (c.f. R4, R6, Section 6.2.1). This was only partially possible in the improved traditional approach and tedious in the basic traditional approach. A possible improvement to reach ++ would be an interactive display of the changes done to the model, as GitLab limits such display to chronological order.

Multi-disciplinary rework phase was also evaluated positively, compared to both traditional approaches. It was easy to retrace what was changed in the course of the rework, who changed it, and why (c.f. R2, Section 6.2.1). Additionally, *CIA Bot* was centralizing the review status information in the activity log of the merge request to provide the holistic overview throughout the change process. Same as in *Multi-disciplinary review* phase, a possible improvement to reach ++ would be an interactive display of the changes done to the model, as GitLab limits such display to chronological order (c.f. R7, R2, Subsection 6.2.1). In the improved traditional approach, it was also possible, but to a limited extent, by adding comments in the MS Teams Tasks. In the basic traditional approach, it was tedious to provide and maintain the same depth of information.

Change integration phase with the M-CIA method also provided good improvement in the majority of the criteria. Version management was perceived as an improvement, given that the *CIA Bot* managed the automatic patch version increment of the model using the semantic versioning[6]. However, the major and minor version still has to be increased manually, which is also the case in both traditional approaches (c. f. R5, Subsection 6.2.1). Semantic change integration done by *CIA Bot* was a great plus in the M-CIA method, as no semantic integration was possible in both traditional approaches. Syntax change integration was evaluated as average in all three approaches, as the state-of-the-art conflict management mechanism of Microsoft Office and Git was used (c.f. R5, Subsection 6.2.1). Traceability of the implementation was also positive, compared to the traditional approaches, as for each change request, it was clear what has changed, when, and by whom to fulfill it (c.f. R2, Subsection 6.2.1). The source of information for this was the Git commit messages and the Git branch. In both traditional approaches, the artifact

[6]Semantic versioning: https://semver.org/

history in Microsoft Office shows who changed the document and when. Still, the message and link to the change request are missing, and it is only visible if two versions of the artifact are compared explicitly. The traceability of changes to the artifact was evaluated similarly to the previous criterion. Still, in the traditional approaches, there is an average capability to trace changes made to one artifact (c.f. R2 Subsection 6.2.1). The CIA method provided a significant improvement in this aspect, as it is possible to see the author and the reason (Git commit message) of each line using the Git blame command.

7.4 EQ3: Feasibility of the M-CIA Method in Batch Production

We have conducted the feasibility study as part of the solution approach description in Section 6.4. We learned that the solution approach is well suited for modeling the discrete production process, as expected, given the results of [Meixner et al., 2021b]. Additionally, we selected a batch production use case for the evaluation of the solution approach to evaluate the feasibility of the approach in such a setting.

We learned that the majority of the aspects from the batch production process use case can be well modeled using the MDM-CPPS DSL. However, given the nature of the batch processing, the assets would contain many more attributes to model the system accurately.

An example of that would be the required production volume of the fertilizer. For that, each of the resources that pump the input products for a process has to know the required volume of the liquid or solid input product. The type of pump has to support the matter state and chemical characteristics of the input product. The assets also need the attribute that would depict the current volume of the input product, and additionally, for each asset, we have to document the maximal allowed volume. For each of those attributes, a dependency on the physical characteristics of the asset needs to be added, such as the weight capacity for measuring the weight of the input product, which must be bigger than the required production volume. The pump and hose material has to be resistant to certain chemicals used in the process, depending on what input product flows in it.

In summary, the approach is feasible for discrete production processes. However, with additional modeling overhead and more complexity, given that each of the resources contains a liquid or solid product, it has to be accurately depicted in the MDEG.

CHAPTER 8

Discussion

This chapter first discusses the observations and lessons learned during the thesis project concerning the research questions in Section 8.1. Afterwards, the chapter describes the limitations of the approach as well as the threats to validity in Section 8.2.

8.1 Observations and Lessons Learned

To validate the proposed M-CIA method and the M-CIA system design, the illustrative use case was analyzed in collaboration with the industry partners from CDL-SQI and the research group members at the TU Wien.

The illustrative use case guided the design of the method for multi-domain CIA. This use case represented a situation in the automotive industry, concretely in the process of screwing car parts together, using robot work cells (c.f. Chapter 5). To engineer a CPPS that assembles cars, engineers from multiple domains work together. These engineers usually work in information silos, often without having an understanding of the system dependencies between various system assets or the whole picture of the production knowledge. The production knowledge is often scattered or implicit. Each engineering domain has its data artifacts in domain-specific formats, which poses high integration complexity. Therefore, CIA in multi-disciplinary CPPS engineering is often inefficient and expensive.

To address the changes, this thesis proposes an improved approach to multi-domain CIA. The M-CIA method aims to foster parallel engineering activities in agile CPPS engineering by identifying the impact of changes in the source domain on related domains and consequently aims to foster multi-domain change impact review coordination.

M-CIA method provides the following capabilities:

- Identification of related stakeholders for impact analysis review

- Transparent and traceable multi-domain CIA coordination
- Change dependency exploration between operational and engineering aspects throughout the CPPS lifecycle

The thesis follows the *Design science methodology* and consults the related work, the industry partners of TheCDL-SQI, and the industry experts (via our expert survey) to understand the requirements for the M-CIA method and an appropriate system design.

For the scope of the thesis, we limited the number of domains that we represent in the illustrative and evaluation use case, as well as the size of the MDEG. Additionally, the evaluation use case differed in the process type from the illustrative use case, as fertilizer production is classified as the batch production process. At the same time, car assembly is a discrete production process.

We also had the opportunity to elicit dependencies not only on the engineering level but also on how the production or operation setup influences the configuration of the CPPS (e.g., if the required production volume of the fertilizer is increased in exceptional cases, the system parts have to be exchanged to handle the bigger volume of the output products). To come to the summarized observations, the first research question was:

RQ1: What is the current state of CIA in multi-domain cyber-physical production systems engineering? The majority of the respondents say that they do not have satisfactory tool support for CIA and they need a better option (cf. Section 6.1). Also, multi-domain changes seem to be prevalent to single-domain changes, which motivates the need for multi-domain CIA methods. Only 20% of the respondents said that their change management tool supports the identification of change impact. The respondents also say that they conduct the CIA mostly manually, based on patterns and previous experience.

We conclude that a formal definition of value dependencies is, therefore, feasible, as they already have the experience and patterns they observe within a review process if a change is implemented. The majority of respondents agree that it is rather easy to identify correct stakeholders within their domain, but it is rather hard to do so in other domains.

As the M-CIA method is based on the MvCM process, proposed by Rinker et al. [2022], we have asked several questions to elicit whether the respondents consider the Git-based approach feasible for their organization (e.g., whether they use source code versioning systems, conduct peer reviews, and whether their systems are stable to formalize them using a domain-specific language). The results were positive; the majority of the respondents stated that they would welcome automated CIA based on a formalized system model. We concluded that the solution approach was feasible and that there still is a need for a tool-supported multi-domain CIA. This motivates the following research question:

RQ2: What methods from agile Software Engineering facilitate applying Git Workflow for efficient multi-domain CIA in CPPS engineering? This research question aimed to understand how the methods from agile Software Engineering, especially Git Workflow, as proposed by Rinker et al. [2022], can facilitate efficient multi-domain CIA. The foundation of the proposed M-CIA method comes from the efficient MvCM based on Git Workflow by Rinker et al. [2022]. We have extended the approach with concrete steps towards automated stakeholder identification, the coordinated impact analysis review process, management of the system dependencies, and the general project setup that enables the integrated Git Workflow, as reported in Rinker et al. [2024].

To assist the CPPS engineers in multi-domain CIA, the proposed method utilizes the concept of a *bot*, which observes the actions of the engineers in the domain-specific models. Based on this information, the *CIA Bot* suggests necessary reviews that include the changing context for the easier review process. Additionally, the bot assigns the review tasks automatically to relevant stakeholders who belong to the impacted domain and have the capacity for the review.

In case the change impact can be propagated to other domain-specific models, the *CIA Bot* does that autonomously. Still, it documents its actions for easy reverting by a human reviewer in case the propagation is unwanted in very specific cases. Furthermore, the bot centralizes the status of the review tasks and updates them to provide transparency in the process. Finally, the bot also performs commits to the common model to integrate updated domain-specific models.

We have compared the approach to two derivations of a traditional approach. The basic traditional approach included working with *Microsoft SharePoint* and asynchronous communication via *Microsoft Outlook*. The improved traditional approach included work with *Microsoft SharePoint* for online collaboration, with the improvement of *Microsoft Teams* for team coordination.

In the quantitative evaluation, we found that the M-CIA method is generally more efficient in terms of execution time, based on the KLM evaluation. In exceptional cases, the method would take more time to execute. However, this is due to the enforcement of data recording, which facilitated traceability. The traceability was perceived as beneficial in the qualitative evaluation. The qualitative evaluation was performed by a basic planner who filled out a Likert scale based on evaluation criteria derived from the requirements. Additionally, we learned that the MDEG created during the case study and its corresponding knowledge graph in combination with the M-CIA method has the potential to break information silos between operation and engineering stakeholders, as envisioned in the DevOps cultural philosophy (c.f. Section 3.5).

In summary, the tools used in traditional CPPS engineering are simpler and easier to use for new users, while using the proposed method requires special training with Git-based repositories or modeling language. However, with more complexity in the system and more stakeholders, office tools provide limited support for traceability, as we saw during the evaluation.

The requirements we specified for the method are technology-agnostic and could be implemented with various technology stacks, e.g., using *Microsoft Office*. The method was based on Rinker et al. [2023b] and Rinker et al. [2022], which gave a great basis for the design of the system architecture, but was extended and integrated to the state-of-the-art tool setup for agile coordination. This leads to the last research question:

RQ3: What system design and architecture can efficiently facilitate conducting the Git Workflow-based multi-domain CIA method? The system design and architecture correspond to the state-of-the-art architectural guidelines, such as SOA, and use a modern technology stack. Architectural concerns such as portability, maintainability, extensibility, and usability were especially addressed. To ensure portability, our solution is containerized to run in a Docker container. To ensure maintainability, the solution is modularized to reduce the technical complexity, as each module represents a different aspect of the system design.

The solution is built in a way that supports connection to Git-based repositories, issue trackers, and integration servers as long as they provide RESTful API and webhooks. Finally, we made use of a state-of-the-art issue tracker with a well-tested user interface to ensure usability. The selection of the Git-based repository and issue-tracking software was based on the reproducibility aspect, which is especially fulfilled with open-source or community edition software. The evaluation with the basic planer from the fertilizer industry showed that the solution has potential, especially with the selected tool setup, given wide adoption in software engineering. However, special training of relevant stakeholders has to be considered before launching such a system on a larger scale in a CPPS engineering organization to ensure proper usage.

8.2 Limitations and Threats to Validity

This section discusses the limitations and threats to the validity of the proposed approach. We classify the threats to validity according to Wohlin et al. [2012].

Changes in the context of properties. A limitation of the current solution approach is that it works only with the property changes of assets in a CPPS. The method does not yet consider structural changes of a CPPS. While value changes are also a great source of change impact on other system parts, structural changes (removal or addition of an asset) are inevitable throughout the CPPS lifecycle. We worked with the assumption that an exchange of an asset for a new one could be simulated by renaming the existing asset and changing the property values. However, neither the removal nor addition of a system part can be accurately represented by this approach. This might lead to the threat of construct validity, as a structural type of change is underrepresented, hence introducing *mono-operation bias*, according to Wohlin et al. [2012].

Small scale use case with simplified dataset. To accurately model any CPPS and evaluate the method in more detail, extensive domain knowledge is necessary to represent the CPPS in the MDM-CPPS DSL. Additionally, we have simplified the dataset only to include several domains. However, as described in Chapter 5, engineers from up to 15 domains work in a car assembly plant. Furthermore, the expert survey results showed that there are stakeholders from other domains, such as data science, software engineering, and or business analysis, for whom the changes in the CPPS are relevant but were not considered as domains in the initial use cases.

This might cause a threat to internal validity due to *interaction of setting and treatment*. Wohlin et al. [2012] define this type of threat as an evaluation of the solution to a simple problem. The case study was conducted with an industry partner from the fertilizer production industry represented by their basic planner, and the evaluation was conducted in an experimental setting. Given that the solution was evaluated in a workshop with one basic planner, *selection* threat to validity might be introduced. We suggest building on the preliminary positive results and evaluating the solution approach for real use in an enterprise with a bigger group of stakeholders as the evaluation participants.

A-priori knowledge representation. Another limitation of the approach is the *a priori* knowledge representation. For the proposed method to work, it is necessary to define the assets and their dependencies a-priori. The algorithm currently identifies explicit dependencies, and support for implicit is not yet covered. This could be done using SWT and machine learning in the future.

Limited variety of use cases. To evaluate the proposed method and system design, we elicited use cases from the automotive and fertilizer production industries. Although the preliminary results are positive, further evaluation of use cases from other industries should be conducted. This might introduce *low statistical power* threat to the conclusion validity [Wohlin et al., 2012].

Evaluation flows of traditional approaches. The preliminary results were positive compared to reconstructed traditional approaches. We reconstructed these approaches as objectively as possible, not to put the traditional approaches to a disadvantage by selecting an expensive evaluation set of steps. However, these reconstructed approaches might differ from the actual multi-domain coordination for CIA in real enterprises. This could introduce a threat to internal validity.

Documentation and user base. Another limitation of the solution approach is documentation and user base, which might impact the solution adoption. The traditional approaches were implemented using *Microsoft Office*, which has a huge community and might, therefore, be easier to learn or troubleshoot. While GitLab, the integrated tool for issue tracking, source code integration, and Git-based repository that we selected, also has a huge community with a lot of helpful documentation, our extensions to GitLab,

the *CIA Bot*, and its usage might not be as obvious to the general engineering audience. This could make the adaption of our solution approach challenging.

Explicit assignment to a discipline is challenging. The current solution approach expects the CPPS to be described using MDM-CPPS DSL and system concepts classified to the domains, which can be perceived as a limitation. However, it is not always easy to define what asset belongs to what domain. The solution for this problem would be defining the concept in every related domain and defining propagation links between the properties. However, this could pose a risk if the propagation links are not maintained or managed properly.

Propagation conflict management. Finally, we acknowledge the conflict management of the automatic propagation as a current limitation. Currently, in the value change propagation scenario, if multiple property value changes propagate to the same property of an asset, the last propagation is going to be the *final*. This might not always be the correct solution to the propagation conflict. To handle such a situation, an appropriate conflict-handling mechanism or a call for human intervention should be implemented.

CHAPTER 9

Conclusion

This chapter concludes the thesis and summarizes its results and contributions. Finally, the chapter describes potential future work topics.

9.1 Conclusion

The main goal of this thesis was to provide an efficient alternative to the traditional coordination approach to multi-domain CIA in CPPS engineering. The previous work of Rinker et al. [2022], their proposed MvCM based on the Git Workflow, was extended, and the M-CIA method was designed to elaborate on the *multidisciplinary impact analysis* step of the MvCM.

The thesis presented two use cases, one illustrative use case from the automotive industry, which depicts a discrete production process based on the extensive domain analysis conducted by the members of the research group at the TU Wien and industry partners of CDL-SQI. The second use case, which represents a batch production process and depicts the interplay between engineering and operation aspects of the CPPS, was defined together with a fertilizer production company.

Additionally, the thesis presented the state-of-the-art (multi-domain) CIA in the literature (literature review) and the industry (expert survey). Based on the literature review, the expert survey, and feedback discussions with our industry partners, the thesis presented the requirements for a multi-domain CIA method and corresponding system design and architecture. To evaluate the M-CIA method, the M-CIA system design was proposed. Based on the system design, the thesis presented a system prototype using a modern technology stack and evaluated the feasibility of the method and the system design.

Furthermore, the thesis presented the results of the solution approach evaluation in a case study with the fertilizer production company by estimating the execution time of representative tasks that are part of an engineering change management process (c.f.

Chapter 5, Figure 5.3). The preliminary results were positive and demonstrated the efficiency of the solution approach. Finally, we evaluated the perceived improvement of the multi-domain CIA method, compared to traditional approaches, in a workshop together with the fertilizer production company representative (basic planner).

In the following, we summarize the results of this work, as depicted in Figure 4.1 for the methodological approach based on Design Science [Hevner et al., 2004]:

Result 1.1 Use case for multi-domain CIA. This result is the illustrative use case *Fasten Screw and Measure with a Robot Cell* defined in Chapter 5 and the evaluation use case *Fertilizer Mixing* defined in Chapter 7.

Result 1.2 State-of-the-art analysis of multi-domain CIA. This result contains findings from the expert survey described in Section 6.1, and findings from the literature review in Chapter 3, and Chapter 2.

Result 2.1 Evaluated M-CIA method. We have elicited seven requirements for a multi-domain CIA method and summarized them in Subsection 6.2.1. Based on these requirements, we designed the M-CIA method and presented it in Section 6.2. Finally, we evaluated the efficiency and perceived improvement of the method with a basic planner from the fertilizer production industry and summarized our findings in Chapter 7.

Result 3.1 System design and architecture. Based on the requirements defined in Subsection 6.2.1 and the method design, we proposed a system design and architecture and documented it in Section 6.3.

Result 3.2 System prototype. We implemented a prototype of a M-CIA management system, following the system architecture from Result 3.1. The detailed documentation of the prototype implementation is available in Section 6.4.

Result 3.3 Evaluation of the system prototype. The system prototype was implicitly evaluated as part of the method evaluation (Result 2.1), but we additionally described the feasibility of the prototype in Section 6.4.

Additionally, in the course of this work, we published a work-in-progress conference paper to evaluate the soundness of our approach in this thesis project[Rinker et al., 2023a]. Parts of the thesis are based on the technical report we published in the course of the work on the thesis project [Rinker et al., 2024], which includes the preliminary method description.

To conclude the thesis, these results contribute to the information system and CPPS community artifacts, knowledge, and insights on: 1) the use-case and requirements for multi-domain CIA in CPPS field, 2) the method for multi-domain CIA, and 3) technology-agnostic information system design for multi-domain CIA within the context of CPPS engineering.

To adopt the results in practice, training to use Git and GitLab, additional software development and extensions to the system design to address the prototype's limitations, modeling of the system in an enterprise setting, and further research is necessary. We define the future work required to achieve the desired impact in the next section.

9.2 Future Work

We specify potential future work based on the limitations described in Chapter 6 and 8. Chapter 6 outlines limitations and threats to the validity of the expert survey results. Chapter 8 outlines limitations and threats to the validity of the method and the system design.

Conduct a more detailed expert survey. As we described previously, the preliminary results of the expert survey to elicit the state-of-the-art practice of M-CIA in industry and research were positive. However, due to limitations defined in Chapter 6, we acknowledged several threats to validity. Therefore, one potential future work activity would be to build on the current version of the questionnaire and conduct a more focused survey in the form of interviews with a bigger participant sample that is narrowed down to specific industries or stakeholder groups to eliminate the validity threats.

Extension of the method for structural changes support and propagation conflict management. The current method, system design, and architecture support changes to attribute values only. One potential future work topic is extending the current results for this aspect. This extension would allow us to depict the change use cases in the industry more accurately, as structural changes are inevitable in CPPS engineering. Additionally, the current version of the method and prototype has basic propagation conflict management in place. To make the method more beneficial in an enterprise setting, the propagation collision mechanism is a potential future work activity.

Sophisticated multi-domain CIA. The solution approach identifies the impact of a change implicitly, which means that if a specific semantic link is not formally defined in the system model, the dependency is not recognized. The contribution of the thesis is a foundation for more sophisticated AI-driven approaches to multi-domain CIA, as it provides a method and system design for gathering validated and well-structured data. Data are crucial for sophisticated approaches such as machine learning. Furthermore, the proposed solution approach envisions the *CIA Bot*, which is currently observing the actions of the CPPS stakeholders and performs relevant actions to coordinate the CIA. However, a potential future work activity could be making this bot more interactive and allow querying the production knowledge and the knowledge that could facilitate the multi-domain CIA in human language, similar to ChatGPT [1].

[1] ChatGPT: `https://chat.openai.com/`

Validation of the method in an enterprise setting. Our evaluation delivered positive preliminary results; however, due to the limitations and threats to validity defined in Chapter 7, a potential future work topic is to apply the method in an enterprise setting to derive additional learnings and improve the method. We expect that with evaluation in an enterprise setting, uncovered limitations, feedback, or improvement potential would be vital to the further design and validation of the M-CIA method.

Possibility to derive digital twins to facilitate digital transformation. Given the validated and well-structured data that result from the M-CIA method and the proposed system design, industry practitioners and research communities could derive digital twins to facilitate automation, integration, and optimization of the production process and to enable monitoring of the operation processes in real-time, to notify relevant stakeholders, if manual intervention during the system operation is necessary [Ugarte Querejeta et al., 2020]. Digital twins are emerging as the latest trend in digital transformation [Ugarte Querejeta et al., 2020], and our results could contribute to further research activities in the field.

List of Figures

List of Tables

Acronyms

AML AutomationML. 27, 43

BPMN Business Process Model and Notation. 27, 28, 41, 91, 92, 130

CC Common Concept. 11–13, 17, 28, 41

CCG Common Concepts Glossary. 11, 12, 74, 83

CDL-SQI Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle. 4, 94, 119, 120, 125

CG Concept Glossary. 11, 83

CGs Concept Glossaries. 10

CIA Change Impact Analysis. 2, 4–7, 9, 15–19, 28–32, 35–42, 47, 48, 51, 52, 54–56, 58–63, 65, 67, 69, 71, 77, 78, 85–88, 94, 115–127, 129–131

CPPS Cyber-Physical Production System. xi, xiii, 1–6, 9–18, 21–23, 25–31, 33–41, 43, 44, 48, 53, 62, 67, 69, 71, 73, 74, 77, 79, 81, 90, 91, 93, 119–127, 129

CPS Cyber-Physical System. 1, 25, 28

DSL Domain-specific Language. 13, 14, 27, 29, 39, 73–75, 94, 118, 123, 124

EMF Eclipse Modeling Framework. 71

IDE Integrated Development Environment. 9, 27, 73, 74, 78, 81

IS Information System. 38, 71

KLM Key-Stroke Level Model. 93, 95–97, 99–111, 113, 121, 131

LSP Language Server Protocol. 73, 74

M-CIA Multi-domain Change Impact Analysis. xi, xiii, xiv, 4–6, 9, 16, 17, 19, 21, 26, 35, 51, 69–71, 78, 81, 82, 86, 89–91, 93, 118–121, 125, 127, 129, 130

M-CIA Multi-domain Change Impact Analysis. xiv, 5, 19, 40, 60, 61, 63, 65, 67, 69, 71, 73, 75, 77–79, 81, 83, 85, 87, 93–97, 99, 101, 103, 105, 107, 109, 111–117, 119–121, 125, 126, 128, 130, 131

MDEG Multi-Domain Engineering Graph. 27, 29, 41, 42, 45, 46, 74, 75, 77, 83, 91, 93, 113, 118, 120, 121, 129, 130

MDM Multi-Domain Modeling. 10, 61

MDM-CPPS Multi-Domain Modeling for CPPS. 9, 10, 13, 14, 27, 62, 71, 73–75, 77, 78, 81, 94, 101, 118, 123, 124, 129

MOF Meta Object Factory. 14, 129

MvCM Multi-view Change Management. 14, 58–60, 65, 71, 120, 121, 125, 130

MvMF Multi-view Modeling Framework. 71

MvMT Multi-view Model Transformation. 60

PAN Product-Process-Resource Asset Network. 27, 39

PDA project-dependent activity. 16–19, 35, 40–42

PIA project-independent activity. 16, 17, 19, 35, 40, 41

PPR Product-Process-Resource. 13, 14, 27–29, 39, 74–76, 79, 80

RDF Resource Description Framework. 22

SOA Service-oriented architecture. 77, 122

SUM Single Underlying Model. 12

SUMI Software Usability Measurement Inventory. 114

SWT Semantic Web Technologies. 28, 29, 123

SysML Systems Modeling Language. 27, 33, 43

TMvMT Traceable Multi-view Model Transformation. 13, 66, 73–75

UML Unified Modeling Language. 27, 29

Bibliography

Don S. Batory and Najd Altoyan. Aocl : A Pure-Java Constraint and Transformation Language for MDE. In *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD,*, pages 319–327, Setúbal, Portugal, 2020. SCITEPRESS. doi: 10.5220/0008942803190327.

Harald Bauer, Alexander Schoonmann, and Gunther Reinhart. Approach for model-based change impact analysis in factory systems. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7, 2017. doi: 10.1109/SysEng.2017.8088301.

Florian Beetz and Simon Harrer. Gitops: The evolution of devops? *IEEE Software*, 39 (4):70–75, 2022. doi: 10.1109/MS.2021.3119106.

Stefan Biffl and Marta Sabou. *Semantic Web Technologies for Intelligent Engineering Applications*. Springer International Publishing Switzerland, 2016. doi: 10.1007/978-3-319-41490-4.

Stefan Biffl, Arndt Lüder, Felix Rinker, Laura Waltersdorfer, and Dietmar Winkler. *Engineering Data Logistics for Agile Automation Systems Engineering*, pages 187–225. Springer International Publishing, Cham, 2019. ISBN 978-3-030-25312-7. doi: 10.1007/978-3-030-25312-7_8.

Stefan Biffl, Juergen Musil, Angelika Musil, Kristof Meixner, Arndt Lüder, Felix Rinker, Danny Weyns, and Dietmar Winkler. An industry 4.0 asset-based coordination artifact for production systems engineering. In *2021 IEEE 23rd Conference on Business Informatics (CBI)*, volume 01, pages 92–101, 2021. doi: 10.1109/CBI52690.2021.00020.

Shawn A. Bohner and Robert S. Arnold. *Software Change Impact Analysis*, chapter An Introduction to Software Change Impact Analysis, pages 1–26. 1996.

Hugo Bruneliere, Jokin Garcia Perez, Manuel Wimmer, and Jordi Cabot. Emf views: A view mechanism for integrating heterogeneous models. In *International Conference on Conceptual Modeling*, pages 317–325. Springer, 2015.

Alistair Cockburn. *Writing Effective Use Cases*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 2000. ISBN 0201702258.

Valerio Cosentino, Javier L. Cánovas Izquierdo, and Jordi Cabot. A systematic mapping study of software development with github. *IEEE Access*, 5:7173–7192, 2017. doi: 10.1109/ACCESS.2017.2682323.

Gwendal Daniel, Gerson Sunyé, and Jordi Cabot. Umltographdb: Mapping conceptual schemas to graph databases. In *International Conference on Conceptual Modeling*, 2016. URL `https://api.semanticscholar.org/CorpusID:4685802`.

Matthias Eckhart, Andreas Ekelhart, Arndt Lüder, Stefan Biffl, and Edgar Weippl. Security development lifecycle for cyber-physical production systems. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 3004–3011, 2019. doi: 10.1109/IECON.2019.8927590.

Mahmoud M. El Nemr and Doaa S. Elzanfaly. A framework for advancing change impact analysis in software development using graph database. In *2018 International Conference on Computer and Applications (ICCA)*, pages 1–438, 2018. doi: 10.1109/COMAPP.2018.8460394.

Hoda Elmaraghy. Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems*, 17:261–276, 01 2005. doi: 10.1007/s10696-006-9028-7.

John Erickson and Keng Siau. Service oriented architecture: A research review from the software and applications perspective. *Innovations in Information Systems Modeling: Methods and Best Practices*, pages 190–203, 01 2009. doi: 10.4018/978-1-60566-278-7.ch010.

Alberto Falcone, Alfredo Garro, Andrea D'Ambrogio, and Andrea Giglio. Engineering systems by combining bpmn and hla-based distributed simulation. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–6, 2017. doi: 10.1109/SysEng.2017.8088302.

Kevin Feichtinger, Kristof Meixner, Felix Rinker, István Koren, Holger Eichelberger, Tonja Heinemann, Jörg Holtmann, Marco Konersmann, Judith Michael, Eva-Maria Neumann, Jérôme Pfeiffer, Rick Rabiser, Matthias Riebisch, and Klaus 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)*, page 1–8. IEEE Press, 2022. doi: 10.1109/ETFA52439.2022.9921568.

Emmanuel Francalanza, Jonathan Borg, and Carmen Constantinescu. A knowledge-based tool for designing cyber-physical production systems. *Computers in Industry*, 84:39–58, 2017a. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2016.08.001.

Emmanuel Francalanza, Jonathan Borg, and Carmen Constantinescu. A computational tool for virtual product development exploiting changeability knowledge. In *21st International Conference on Engineering Design*, volume 4, pages 593–602, 2017b.

Erik Frøkjær, Morten Hertzum, and Kasper Hornbæk. Measuring usability: are effectiveness, efficiency, and satisfaction really correlated? *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 2000. URL https://api.semanticscholar.org/CorpusID:328946.

Irlán Grangel-González, Felix Lösch, and Anees ul Mehdi. Knowledge graphs for efficient integration and access of manufacturing data. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 93–100, 2020. doi: 10.1109/ETFA46521.2020.9212156.

Lavdim Halilaj, Irlán Grangel-González, Gökhan Coskun, Steffen Lohmann, and Sören Auer. Git4voc: Collaborative vocabulary development based on git. *Int. J. Semantic Comput.*, 10(2):167–192, 2016. doi: 10.1142/S1793351X16400067.

Csaba Hegedüs, Pál Varga, and Attila Frankó. A devops approach for cyber-physical system-of-systems engineering through arrowhead. *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 902–907, 2021.

Robert Heinrich, Kiana Busch, and Sandro Koch. A methodology for domain-spanning change impact analysis. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 326–330, 2018. doi: 10.1109/SEAA.2018.00060.

Alan R. Hevner. A three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19:4, 2007.

Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004. ISSN 02767783. URL http://www.jstor.org/stable/25148625.

Constantin Hildebrandt, Aljosha Köcher, Christof Küstner, Carlos-Manuel López-Enríquez, Andreas W. Müller, Birte Caesar, Claas Steffen Gundlach, and Alexander Fay. Ontology building for cyber–physical systems: Application in the manufacturing domain. *IEEE Transactions on Automation Science and Engineering*, 17(3):1266–1282, 2020. doi: 10.1109/TASE.2020.2991777.

Xuan Luu Hoang, Alexander Fay, Philipp Marks, and Michael Weyrich. Industrial application of a mdm-based approach for generation and impact analysis of adaptation options - a case study. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1244–1247, 2018. doi: 10.1109/ETFA.2018.8502460.

David E. Kieras. Using the keystroke-level model to estimate execution times. *Technical Report*, 2003. URL https://api.semanticscholar.org/CorpusID:59918479.

J. Koch, A. Gritsch, and G. Reinhart. Process design for the management of changes in manufacturing: Toward a manufacturing change management process. *CIRP Journal of Manufacturing Science and Technology*, 14:10–19, 2016. ISSN 1755-5817.

István Koren, Felix Rinker, Kristof Meixner, Jasminka Matevska, and Jörg Walter. Challenges and opportunities of devops in cyber-physical production systems engineering. In *2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 1–6, 2023. doi: 10.1109/ICPS58381.2023.10128073.

Andreas Kreutz, Gereon Weiss, Johannes Rothe, and Moritz Tenorth. Devops for developing cyber-physical systems. Technical report, Fraunhofer Institute for Cognitive Systems IKS, 2021.

Steffen Lehnert. A review of software change impact analysis. 2011a. URL `https://api.semanticscholar.org/CorpusID:10622808`.

Steffen Lehnert. A taxonomy for software change impact analysis. In *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution*, IWPSE-EVOL '11, page 41–50, New York, NY, USA, 2011b. Association for Computing Machinery. ISBN 9781450308489. doi: 10.1145/2024445.2024454.

Bixin Li, Xiaobing Sun, Hareton Leung, and Sai Zhang. A survey of code-based change impact analysis techniques. *Software Testing, Verification and Reliability*, 23, 12 2013. doi: 10.1002/stvr.1475.

Miles Macleod, Rosemary Bowden, Nigel Bevan, and Ian Curson. The music performance measurement method. *Behaviour and Information Technology*, 16, 07 1997. doi: 10.1080/014492997119842.

Kristof Meixner, Arndt Lüder, Jan Herzog, Dietmar Winkler, and Stefan Biffl. Patterns for reuse in production systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 31:1623–1659, 12 2021a. doi: 10.1142/S0218194021400155.

Kristof Meixner, Felix Rinker, Hannes Marcher, Jakob Decker, and Stefan Biffl. A Domain-Specific Language for Product-Process-Resource Modeling. In *26th IEEE ETFA*, 2021b.

Kristof Meixner, Felix Rinker, Laura Waltersdorfer, Arndt Lüder, and Stefan Biffl. Organizing reuse for production systems engineering with capabilities and skills. *Autom.*, 71(2):116–127, 2023.

Maximilian Meißner, Georg Jacobs, Patrick Jagla, and Jonathan Sprehe. Model based systems engineering as enabler for rapid engineering change management. *Procedia CIRP*, 100:61–66, 2021. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2021.05.010. 31st CIRP Design Conference 2021 (CIRP Design 2021).

Alachew Mengist, Lena Buffoni, and Adrian Pop. An integrated framework for traceability and impact analysis in requirements verification of cyber–physical systems. *Electronics*, 10(8), 2021. ISSN 2079-9292. doi: 10.3390/electronics10080983.

László Monostori. Cyber-physical production systems: Roots, expectations and r&d challenges. *Procedia CIRP*, 17:9–13, 2014. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2014.03.115.

Cezary Orłowski, Artur Ziółkowski, Aleksander Orlowski, Pawel Kaplanski, Tomasz Sitek, and Witold Pokrzywnicki. *Ontology of the Design Pattern Language for Smart Cities Systems*, volume 9990, pages 76–100. 09 2016. ISBN 978-3-662-53579-0. doi: 10.1007/978-3-662-53580-6_6.

Laurin Prenzel and Sebastian Steinhorst. Decentralized autonomous architecture for resilient cyber-physical production systems. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1300–1303, 2021. doi: 10.23919/DATE51398.2021.9473954.

Cosmina Cristina Raţiu, Wesley K. G. Assunção, Rainer Haas, and Alexander Egyed. Reactive links across multi-domain engineering models. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, MODELS '22, page 76–86, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450394666. doi: 10.1145/3550355.3552446.

Felix Rinker. Flexible multi-aspect model integration for cyber-physical production systems engineering. In *Proceedings of Doctoral Consortium Papers Presented at the 33rd International Conference on Advanced Information Systems Engineering (CAiSE 2021)*, volume 2906, pages 31–40, 07 2021.

Felix Rinker, Laura Waltersdorfer, Kristof Meixner, and Stefan Biffl. Towards support of global views on common concepts employing local views. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1686–1689, 2019. doi: 10.1109/ETFA.2019.8869239.

Felix Rinker, Kristof Meixner, Laura Waltersdorfer, Dietmar Winkler, Arndt Luder, and Stefan Biffl. Towards efficient generation of a multi-domain engineering graph with common concepts. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4. IEEE, sep 2021. doi: 10.1109/etfa45728.2021.9613657.

Felix Rinker, Sebastian Kropatschek, Thorsten Steuer, Kristof Meixner, Elmar Kiesling, Arndt Lüder, Dietmar Winkler, and Stefan Biffl. Efficient multi-view change management in agile production systems engineering. In *24th International Conference on Enterprise Information Systems (ICEIS 2022)*, volume 2, pages 134–141, 05 2022. doi: 10.5220/0011074000003179.

Felix Rinker, Diana Vysoká, Kristof Meixner, David Hoffmann, and Stefan Biffl. Organizing multi-domain change impact analysis in cyber-physical production systems engineering. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2023a. doi: 10.1109/ETFA54631.2023.10275684.

Felix Rinker, Laura Waltersdorfer, Kristof Meixner, Dietmar Winkler, Arndt Lüder, and Stefan Biffl. Traceable multi-view model integration: A transformation pipeline for agile production systems engineering. *SN Comput. Sci.*, 4(2):205, 2023b. doi: 10.1007/s42979-022-01572-5.

Felix Rinker, Kristof Meixner, Diana Vysoká, and Stefan Biffl. The MDM-CPPS Framework: GitOps-enabled Multi-Domain Modeling in Cyber-Physical Production Systems Engineering. Technical Report CDL-SQI 2024-01, CDL-SQI, Inst. for Information Systems Eng., TU Wien, jan 2024.

S Roopa and Rani Menta Satya. Questionnaire designing for a survey. *The Journal of Indian Orthodontic Society*, 46:37–41, 06 2012. doi: 10.5005/jp-journals-10021-1104.

Jeff Sauro. Estimating productivity: Composite operators for keystroke level modeling. In Julie A. Jacko, editor, *Human-Computer Interaction. New Trends*, pages 352–361, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-02574-7.

Gail Sullivan and Anthony Artino. Analyzing and interpreting data from likert-type scales. *Journal of graduate medical education*, 5:541–2, 12 2013. doi: 10.4300/JGME-5-4-18.

Jun Tie, Jia Jin, and Xiaorong Wang. Study on application model of three-tiered architecture. In *2011 Second International Conference on Mechanic Automation and Control Engineering*, pages 7715–7718, 2011. doi: 10.1109/MACE.2011.5988838.

Christian Tunjic and Colin Atkinson. Synchronization of projective views on a single-underlying-model. In *Proceedings of the 2015 Joint MORSE/VAO Workshop on Model-Driven Robot Software Engineering and View-based Software-Engineering*, pages 55–58, 2015.

Miriam Ugarte Querejeta, Leire Etxeberria, and Goiuria Sagardui. Towards a devops approach in cyber physical production systems using digital twins. In António Casimiro, Frank Ortmeier, Erwin Schoitsch, Friedemann Bitsch, and Pedro Ferreira, editors, *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops*, pages 205–216, Cham, 2020. Springer International Publishing.

VDI/VDE 3682. VDI/VDE 3682: Formalised process descriptions. Beuth Verlag, 2005.

VDI/VDE 3695. VDI/VDE 3695: Engineering of industrial plants. Beuth Verlag, 2013.

Chengcheng Wan, Zece Zhu, Yuchen Zhang, and Yuting Chen. Multi-perspective change impact analysis using linked data of software engineering. In *Proceedings of the 8th Asia-Pacific Symposium on Internetware*, Internetware '16, page 95–98, New York,

NY, USA, 2016. Association for Computing Machinery. ISBN 9781450348294. doi: 10.1145/2993717.2993729.

Roel Wieringa. Design science methodology for information systems and software engineering. In *Springer Berlin Heidelberg*, 2014. URL `https://api.semanticscholar.org/CorpusID:8371388`.

Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*, chapter 5 Case Studies. Springer-Verlag Berlin Heidelberg, 2012.

Andreas Wortmann, Olivier Barais, Benoit Combemale, and Manuel Wimmer. Modeling languages in Industry 4.0: an extended systematic mapping study. *Software and Systems Modeling*, 19(1):67–94, sep 2020. doi: 10.1007/s10270-019-00757-6.

Xuan Wu, Virginie Goepp, and Ali Siadat. Cyber physical production systems: A review of design and implementation approaches. In *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1588–1592, 2019. doi: 10.1109/IEEM44572.2019.8978654.

W.M. Yang, C.D. Li, Y.H. Chen, and Y.Y. Yu. Change impact analysis of complex product using an improved three-parameter interval grey relation model. *Advances in Production Engineering & Management*, 16:185–198, 06 2021. doi: 10.14743/apem2021.2.393.