TU WIEN Informatics

Multi-Aspect Test Case Specification and Automation based on Cause-Effect Analysis

a Case Study in Industrial Environments

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Serafima Sherstneva, BSc.
Matrikelnummer 01328109

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Ass. Dipl.-Ing. Dr.techn. Dietmar Winkler
Mitwirkung: Ao.Univ.Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl

Wien, 3. Oktober 2022

Serafima Sherstneva	Dietmar Winkler

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

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Informatics

Multi-Aspect Test Case Specification and Automation based on Cause-Effect Analysis

a Case Study in Industrial Environments

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Business Informatics

by

Serafima Sherstneva, BSc.
Registration Number 01328109

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Ass. Dipl.-Ing. Dr.techn. Dietmar Winkler
Assistance: Ao.Univ.Prof. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Biffl

Vienna, 3rd October, 2022

Serafima Sherstneva	Dietmar Winkler

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

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Serafima Sherstneva, 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, 3. Oktober 2022

Serafima Sherstneva

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

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

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Kontext und Motivation. In "Multi-disciplinary Engineering" (MDE) Umgebungen, ist das Wissen über unterschiedliche Aspekte eines System, wie "Product, Process and Resources" (PPR) Spezifikationen, als auch Umweltfaktoren und deren Einflüsse auf das Produkt und den Prozess häufig auf Experten unterschiedlicher Disziplinen verteilt, die divergierende Blickwinkel auf das System haben. Dadurch fällt es oft schwer, dieses Wissen an einem Ort zu konzentrieren. Testen ist eine Methode, um beabsichtigte Effekte zu erforschen, wie zum Beispiel ob das Produkt die erwarteten Anforderungen erfüllt, ob der Produktionsprozess die notwendige Durchsatzleistung erbringt, oder Risiken, die zu unerwünschten Effekten führen, zu überprüfen. Traditionelle Testmethoden betrachten of nur Effekte, die einer bestimmten Disziplin zuzuordnen sind. In MDE's, etwa bei "Cyber-physical Production Systems" (CPPS) ist die Formulierung und Validierung von Ursachen-Wirkung-Hypothesen eine komplexe Aufgabe. Dadurch bleiben Risiken, auch bei der Verwendung von systematischen Testmethoden, oft unentdeckt.

Ziele. Das primäre Ziel dieser Studie ist der Entwurf eines Ansatzes, der die systematische Testplanung, Messung und Analyse in MDE Umgebungen, auf Basis der Ursachen-Wirkung-Analyse, ermöglicht.

Methoden. Basierend auf dem Ziel wurden die folgenden Forschungsfragen formuliert: (1) Welche Risikotreiber motivieren Multi-Aspekt Risikobasiertes Testen von Ursachen-Wirkung-Hypothesen zwischen Eigenschaften des Produktionsprozesses und Parametern/Einflüsse der Produktionsressourcen? Das Subset der Risikotreiber wurde durch den Forschungsüberblick und Workshop-Veranstaltungen mit Fachexperten abgeleitet. (2) Welches Wissensmodell kann Multi-Aspekt Ursachen-Wirkung-Hypothesen zwischen Eigenschaften des Produktionsprozesses und Parametern/Einflüsse der Produktionsressourcen repräsentieren? Aufbauend auf der "Failure Modes and Effects Analysis" (FMEA) zur Unterstützung von systematischem und effizientem Risikomanagement, "PPR Asset Network" (PAN) Metamodell zur Beschreibung einer CPPS Struktur und "CPPS Risk Assessment (CPPS-RA) Metamodell und Methode die FMEA, PAN und Ursachen-Wirkung-Analyse Aspekte verbindet, erweitert um testrelevante Aspekte, wurde das "Multi-aspect Test Case Specification" (MATCS) Metamodell erstellt und fachspezifische Wissensmodelle abgeleitet. (3) Welche Prozessschritte sind zur Unterstützung von Multi-Aspekt-Testung in "CPPS" und Softwaretechnik erforderlich? Basierend auf dem "Test Driven Development" (TDD) Ansatz zur Unterstützung von Software Design, der

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

"Equivalence Class Partitioning" (ECP) Technik zur Aufteilung des Testraumes in Datenklassen und Gherkin Notation zur einfachen repräsentation von automatisieren Tests, wurde MATCS konzipiert. Die Risikoteiber, das MATCS Metamodell und die MATCS Methode repräsentieren in Kombination den MATCS Ansatz. Um die Forschungsfragen zu beantworten, wurde der "Design Science Research" Ansatz angewandt.

Ergebnisse. MATCS führt das Subset an fachspezifischen Risikotreibern, das MATCS Metamodel und Methoden ein. Die Evaluierung des Ansatzes zeigt aussichtsreiche Resultate im Kontext von zwei Fallstudien in den Bereichen von CPPS und Softwaretechnik. Der Ansatz unterstützt die Fachexperten bei der Spezifikation von effizienten Testfällen für die Validierung der Ursachen-Wirkung-Hypothesen, der Definition des Umfangs und Limitationen der Testautomatisierung und macht das Experimentieren für Systemverbesserungen systematischer im Vergleich zu traditionellen Ansätzen. Ein PAN ist die Voraussetzung für eine effiziente Anwendung des Ansatzes.

Fazit. MATCS hilft dabei, das implizite Wissen der multi-disziplinären Fachexperten explizit zu machen und schlägt weitere Schritte für die Qualitätsexperten vor, um Multi-Aspekt Testfälle zu spezifizieren und sowohl Umfang als auch Limitierungen für die Testautomatisierung zu definieren. Die initiale PAN Definition erfordert zwar Mehraufwand, macht sich aber in zukünftigen Iterationen im Kontext einer kontinuierlichen Verbesserungsstrategie bezahlt.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Context and Motivation. In Multi-disciplinary Engineering (MDE) environments, knowledge on different aspects of the system, such as product, process and resources (PPR) specifications, as well as environmental factors, and their influence on the product and process quality is often distributed among experts coming from multiple disciplines and having different views on the system, and has been found hard to concentrate in one place. Testing is a method to explore intended effects, such as whether a product matches expected requirements, a production process meets performance goals, as well as to check for risks, which could cause undesired effects. Traditional testing methods often consider causes responsible for the effects coming from one discipline. However, in MDEs, such as Cyber-physical Production Systems (CPPS) engineering, due to their multi-disciplinary nature, systematic formulation and validation of cause-effect hypotheses is a complex task, therefore risks often remain undiscovered by systematic testing methods.

Objectives. The primary objective of this study was to design an approach to enable systematic test planning, measurement and analysis in MDE environments based on cause-effect analysis.

Methods. Based on the goal, the following research questions were formulated: (1) Which risk drivers motivate multi-aspect risk-based testing of cause-effect hypotheses between production process characteristics and production resource parameters/influences? The sub-set of risk drivers was derived from the literature review and workshops with domain experts. (2) What knowledge model can represent multi-aspect cause-effect hypotheses between production process characteristics and production resource parameters/influences? Built on the Failure Modes and Effects Analysis (FMEA) to support systematic and efficient risk management, PPR Asset Network (PAN) meta-model to describe a CPPS structure, and CPPS Risk Assessment (CPPS-RA) meta-model and method which connects FMEA, PAN and cause-effect analysis aspects, extended with test related aspects, Multi-aspect Test Case Specification (MATCS) meta-model was designed and domain specific knowledge models were derived. (3) What are the process steps to support multi-aspect testing in CPPS and software engineering? Based on the Test Driven Development (TDD) approach to support system design, Equivalence Class Partitioning (ECP) technique to divide the test space into classes of data, and Gherkin notation to allow easy representation of automated tests, MATCS was designed. Risk drivers, MATCS meta-model and MATCS method combined represent the MATCS ap-

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

proach. In order to answer the research questions, the Design Science Research approach, supported by a literature review was followed. MATCS was evaluated in CPPS and software engineering industrial case studies, which included workshops and experiments.

Results. MATCS introduced the sub-sets of domain specific risk drivers, MATCS meta-model and method. The evaluation of the approach has shown promising results in context of two case studies, in CPPS and software engineering domains. The approach supports the domain experts in efficient test case specification for cause-effect hypotheses validation, definition of scope and limitations for test automation and makes experimentation for system improvements more systematic in comparison to the traditional approach. High quality PAN is the prerequisite for efficient application of the approach.

Conclusion. MATCS helps to make the implicit multi-disciplinary expert knowledge explicit and suggests steps to the quality experts allowing them to specify multi-aspect test cases and define the scope and limitations for test automation. The initial PAN definition requires considerable effort, however, it is expected to pay off in future iterations in context of continuous integration strategy.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

Due to the constantly growing complexity of software-intensive production systems [Biffl et al., 2017b], experts in individual disciplines are no longer able to adequately assure the quality of production processes without taking into consideration *aspects* from multiple disciplines, which could contribute to incorrect or unexpected results. Such aspects are multi-domain specific product, process, and resource (PPR) specifications, as well as environmental influences.

Testing is a method to explore *intended effects*, such as whether a product or a process matches expected requirements, a production process meets performance goals, as well as to check for risks, which could cause *undesired effects*. The testing process can start at the requirements definition phase and continue throughout the engineering and production processes. Knowledge on different aspects of the system and their influence on the product quality is often distributed among experts coming from different disciplines and has been found hard to concentrate in one place [Meier et al., 2020, Biffl et al., 2021]. This fact adds even more complexity to the standard test case specification challenges that test engineers meet at their daily work, such as sufficient business and code coverage, as well as resource availability and time limitations [Felderer and Ramler, 2014, Meixner et al., 2020].

In order to reduce testing time and costs, some of the tests can be automated. Test automation plays a crucial role in complex software-intensive production systems due to the large testing scope and limited time frame. Automation of test cases having multi-aspect nature is often limited due to heterogeneous tools, artifacts, and systems. Moreover, software test automation experts need to be involved in the process of test automation as well as experts with domain knowledge [Winkler et al., 2018, Meixner et al., 2020].

Therefore, the process of test case design and automation in multi-aspect engineering environments remains challenging.

This thesis focuses on *testing system behavior controlled by software* that depends on assumptions, requirements, and designs coming from related "other" systems, engineering disciplines and providers.

1.1 Problem Statement

This section aims to present the challenges derived from the two minimal illustrative use cases which coming from the industry partners. Both of the use cases are typical for the organizations. The use cases are typical withing the organizations and represent real scenarios. Based on the identified challenges, research questions are derived. The answers to the research questions aim to support the domain experts in overcoming their challenges.

1.1.1 Minimal illustrative use cases

Two typical domain specific minimal use cases are presented to support understanding of the research problematic.

Minimal illustrative use case *Algorithm performance at Company A.* A typical example of a multi-aspect environment is a large-scale Company *A* with over 20 million registered online customers. One of the products of the company is a Web service delivering data to the front-end application for the end users.

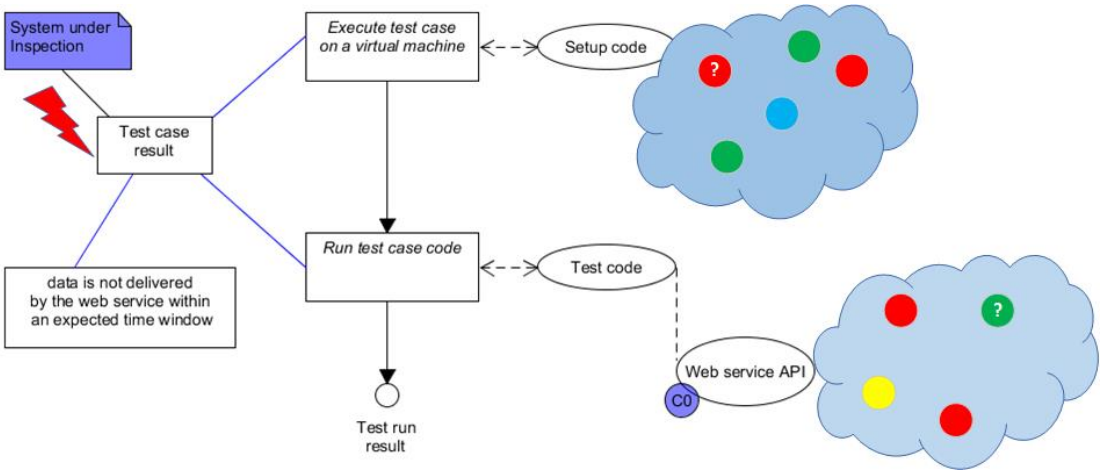

Figure 1.1: Minimal illustrative use case *Algorithm performance at Company A.*

The testing task is to assure that the software algorithm responsible for the data delivery works as expected and the Web service delivers the data within an expected time window. If the data is not delivered by the service within an expected time window, visible delays could appear on the front-end of the application. The testing task is necessary since the response time is the critical measurement for customer satisfaction. In order to save testers' time and effort, a test automation project displayed in Figure 1.1 is in place.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The test case results coming from the automated test execution, therefore, are not only dependent on the performance of the software algorithm, the performance of underlying Web service infrastructure, e.g., a virtual machine where the service is deployed, but also on the performance of the underlying automating system, e.g., a virtual machine where the test case is executed.

Figure 1.1 displays the traditional test automation process with focus on the algorithm (the cause marked as C_0), while in this work *causes* in the underlying automating system and the environment, causes, such as CPU or memory load, which are likely to have an impact on the software algorithm performance (Figure 1.1: the causes on the clouds) are considered as well.

Illustrative use case *4-color printing with Industry 4.0 components.*

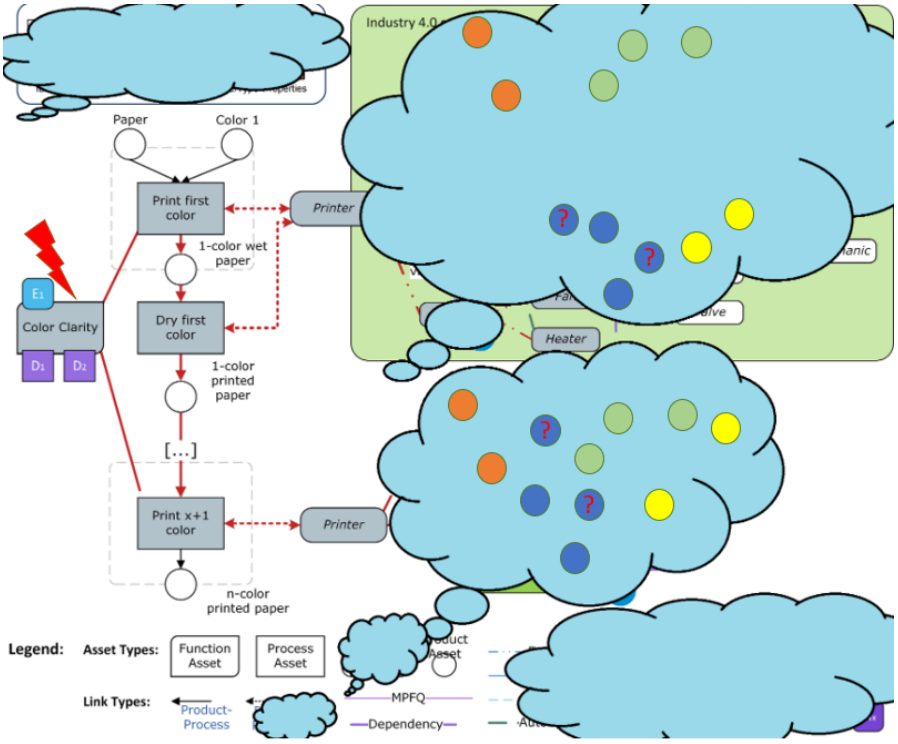

Figure 1.2: Use case *4-color printer with Industry 4.0 components.* Based on [Biffl et al., 2020a].

The use case is identified from the relevant literature and an initial set of requirements was established based on the use case. Figure 1.2 displays the elements of the use case: an industrial four-color printing solution which is achieved by overlaying of four basic colours [Biffl et al., 2020a]. The printing process includes usage of paper and colors, and sequential printing and drying of the layers. In order to build such a printer, 8 to 10 engineering views could be involved. The example illustrates the difficulty to test

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

hypotheses on multi-aspect relationships between the effect, such as insufficient *color clarity*, and possible causes which could trigger the effect.

The causes coming from the underlying systems may go back to natural laws relevant to production systems. These causes are typically not explicitly represented, but implicit parts of assumptions, requirements, and test cases. Due to their implicit nature these causes are not part of systematic planning, measurement, and analysis. Therefore, test case design may be incomplete and based on wrong assumptions and may not measure relevant data to explain why a test case failed as foundation for analyzing the system parts that require improvement.

Shortcomings. Figure 1.3 represents main shortcomings of the test case specification and automation process in a multi-disciplinary environment, such as no efficient representation of implicit domain knowledge, stakeholder dependencies, as well a test case coverage, and automation.

The aim of the process is to ensure that different aspects of the production system contribute to the final product the way they should by providing sufficient test coverage and test automation. The process includes the following steps: identification of possible risk elements of the production system, test case design and data source definition, as well as definition of scope and limitations for automation for the selected test cases.

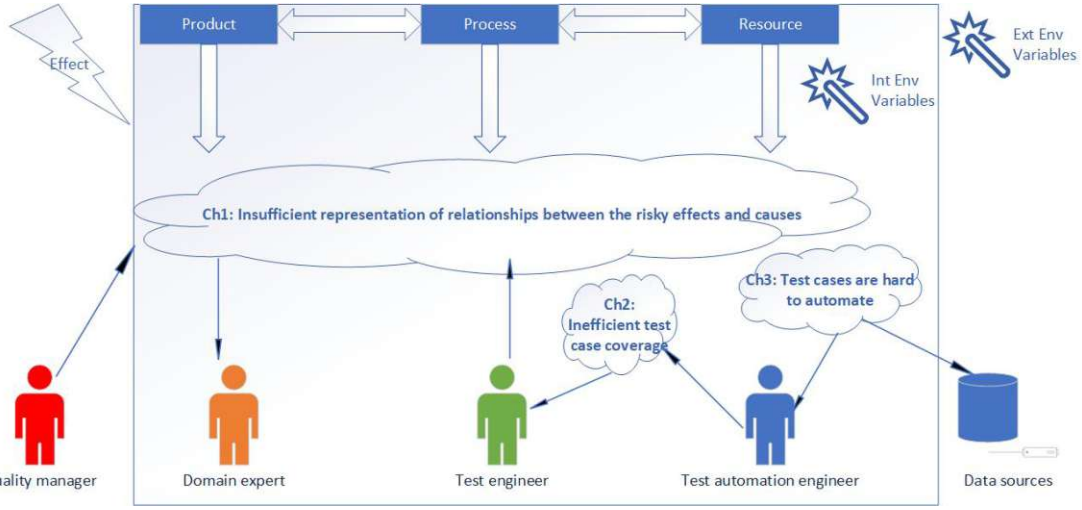

Figure 1.3: Major challenges in test case specification and automation in a multi-disciplinary engineering environment.

There are four main stakeholders, represented as actors in Figure 1.3: the quality manager (in red color), the domain expert (orange), the test engineer (green), and the test automation engineer (blue). The four stakeholders work in a close cooperation and a person may take on more than one role.

The *Quality Manager* is responsible for test planning, resource allocation, and decision

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

making based on test reports [Winkler et al., 2018]. In order for the quality manager to assure the quality of the *System under Test (SuT)*, he needs to make sure that everything that should go right will go right. The correct outputs are *desired effects*. On the other hand, the quality manager also needs to assure that everything that could go wrong with the product will not go wrong. Anything which could go wrong is considered an *undesired effect*. In multi-disciplinary environments, just like in pure software engineering environments, there is a large number of variables that may cause different outcomes and their effects need to be understood [Wohlin et al., 2014]. *A cause* can be an attribute of the domain, of a resource, of the environment, or even a human factor (human factors are out of scope in this work). Furthermore, multiple disciplines are often involved, for example, software engineering, electrical engineering, and mechanical engineering.

Domain experts from multiple disciplines are typically aware of the technical requirements of the system and have knowledge on the effects, implicit and explicit causes, and their relationships. Therefore, the knowledge of domain experts is required for test case design. The assumption that the effect is related to a particular cause or a combination of causes is called *hypothesis* [Wohlin et al., 2014].

Test engineers validate the domain experts' hypotheses by designing test cases that lead to desired effects and cover risks that may lead to undesired effects.

Test automation engineers prepare and maintain the test automation framework, support domain experts and test engineers by implementing code as the basis for the test cases, and make decisions on what can and should be automated in cooperation with domain experts and test engineers.

1.1.2 Challenges

From the minimal illustrative use cases major challenges in *multi-aspect test case specification and automation* are derived:

Challenge Ch1. Insufficient representation of multi-aspect cause-effect hypotheses. *There is currently no efficient way of the multi-aspect cause-effect relationships knowledge representation for selected hypotheses leading to insufficient overview.* Test engineers focus on design and execution of multi-aspect test cases in cooperation with domain experts, in order to validate the hypotheses. In addition to Ch1, further challenge for test engineers and this work is derived.

Challenge Ch2. Inefficient coverage of multi-aspect test cases. *Specified test cases often do not cover all critical parts of the system and its characteristics or the opposite, many test cases validate similar assumptions.* The goal of the test automation engineer is to deliver a robust and stable solution for automation of the multi-aspect test cases with automated data collection from different sources. In multi-engineering environments, test automation engineers meet the following challenge.

Challenge Ch3. Multi-aspect test cases are hard to automate. *Due to the multiple aspects involved in the test automation process, different tools are required to*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

test different artifacts and parts of the system. Moreover, test automation engineers have to address the following decisions: (a) what test cases are possible to automate; (b) what test cases make sense to automate from a cost-benefit perspective; and (c) how to automate possible test cases that make sense to automate.

1.1.3 Research questions

The following research questions have been addressed in the context of multi-aspect testing in software and CPPS engineering by conducting a case study in a company with distributed software system setup and a case study in a CPPS company partner:

- *RQ1: Which risk drivers motivate multi-aspect risk-based testing of cause-effect hypotheses between production process characteristics and production resource parameters/influences?*

  This research question is stated in order to identify sets of risk drivers which motivate multi-aspect testing of system requirements for subsets of software-intensive (production) systems. In order to answer this question, a literature review has been conducted in the first place. Based on the output from the literature review, multiple workshops took place where various domain experts have reviewed and discussed the identified risk drivers and selected the ones which are most relevant for their use cases.

- *RQ2: What knowledge model can represent multi-aspect cause-effect hypotheses between production process characteristics and production resource parameters/influences?*

  Domain experts require a model which supports them in representation of their heterogeneous multi-disciplinary knowledge and building of hypotheses on cause-effect relationships. This research question focuses on a meta-model design which allows to instantiate such domain specific knowledge models. The requirements for the meta-model and the model are established based on the previous work and workshops and discussions with the domain experts. Both, meta-model and its instances are evaluated in the case studies with the company partners.

- *RQ3: What are the process steps to support multi-aspect testing in CPPS and software engineering?*

  The aim of this research question is to design a method which allows test engineers to define a systematic test suite to define a minimal set of tests that collect data on cause-effect relationships to validate the hypotheses. Additionally, the method must include steps which support the domain experts in definitions of the scope and limitation for test automation. The method is evaluated by its application for selected use cases in case studies in cooperation with domain experts.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

1.2 Aim of the Work

From the problem definition and the use cases *Algorithm performance at Company A*, the following requirements for the contributions of this thesis, research questions, and use cases for evaluation are derived. In this work *Design Science* research [Engström et al., 2020, Wieringa, 2014] is employed in order to investigate how to improve shortcomings in the context of production processes automated by (software-intensive) production resources/systems.

This thesis aims at making the causes for selected effects explicit to enable efficient systematic test planning, measurement and analysis. In figure 1.4, the three corresponding contributions to the previously introduced challenges are presented. This figure is structured the same way as the previously introduced figure 1.4.

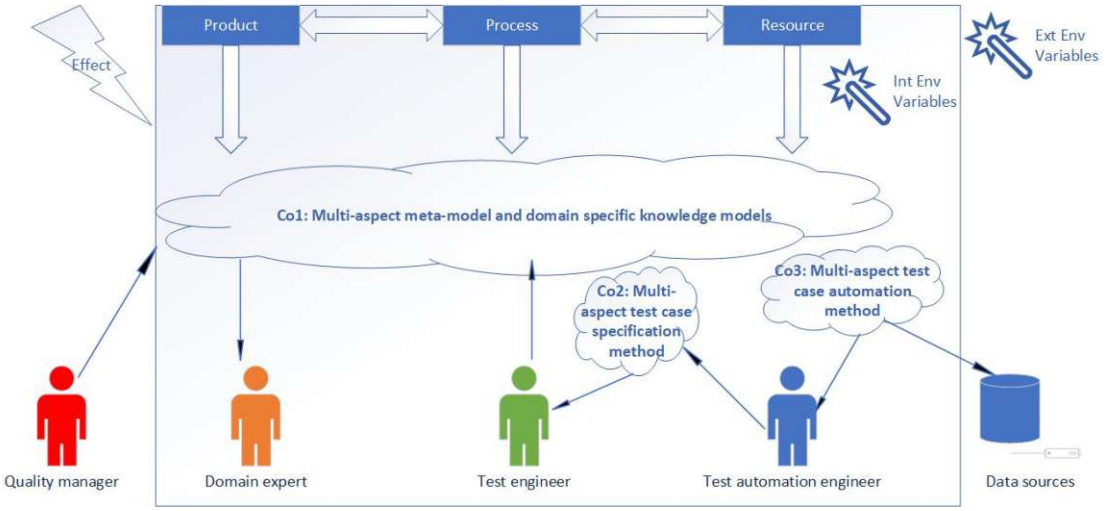

Figure 1.4: Key contributions of this thesis.

Key contributions: The main contributions of this master thesis are derived based on the challenges stated in Section 1.1.2:

- *Co1: Multi-aspect meta-model and domain specific knowledge models*: software engineering knowledge model and CPPS engineering knowledge model. The models support: quality managers, who have to identify multi-aspect factors that are likely to lead to specific quality risks, as a foundation for mitigating these risks; domain experts, who require an efficient method for representing domain knowledge on multi-disciplinary dependencies; quality engineers, who have to plan and execute testing activities in order to validate multi-aspect cause-effect hypotheses; test automation engineers, who have to support test engineers in automating test cases with multi-disciplinary dependencies. The model can be queried and is iteratively expandable.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- *Co2: Multi-aspect test case specification method* which allows the quality experts to specify the minimum amount of test cases saving testing resources without lowering the quality of the results in multi-disciplinary engineering environments.
- *Co3: Multi-aspect test case automation method* which supports test automation engineers in definition of test case automation scope and limitations in multi-disciplinary environments.
- *Use cases*, groups of different data and requirements that highlight characteristics of multi-aspect knowledge in engineering processes.
- *Test-driven systems engineering* that explains which stakeholders are capable and responsible for system behavior design and could provide important models and data for testing.

These contributions are relevant for software-intensive systems roles and scientific communities, such as for instance participants of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)[1]. The method and its conceptual evaluation has been published in the 27th peer-reviewed in conference (IEEE International Conference on ETFA [Winkler et al., 2022].

1.3 Structure of the Work

Chapter 1 introduces the motivation, problem statement with illustrative use cases and aim of the work. Chapter 2 is introduced in order to present the related work. Following, Chapter 3 presents the Research Issues and the appropriate approach to address the issues. In Chapter 4 the main contribution of this work is described: the multi-aspect test case specification method. Chapter 5 is designed to describe the software engineering use case "Algorithm performance" and presents the case study and the results. Chapter 6 follows the same structure Chapter 5 with the focus on the CPPS aluminium surface cleaning process. The comparative analysis of the case study results is depicted, each individual research issue is discussed and possible limitations that the research faces are presented in Chapter 7. Chapter 8 concludes the thesis and presents possible future work.

[1]ETFA: https://ieeexplore.ieee.org/xpl/conhome/1000260/all-proceedings

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Related Work

Chapter 1 introduced the context of the research and key challenges which the stakeholders in Multi-disciplinary Engineering (MDE) environments, such as Cyber-Physical Production Systems (CPPS) Engineering and software engineering meet.

The Related Work chapter provides information on the recent research state about the most important topics of this thesis: MDE, Risk Assessment (RA) and software system testing, for various software-intensive engineering contexts. Section 2.1 summarizes related work on MDE, and specifically the associated with software-intensive (production) systems challenges, and gives an overview about terminology. Following section 2.2 presents the concept of RA for software-driven systems such as CPPS and software systems. Finally, section 2.3 explores software system testing strategies and techniques, as well as the possibilities of test automation in MDE environments.

2.1 Multi-disciplinary Engineering

"Software-intensive systems are systems in which software development and/or integration are dominant considerations" [ISO, 2011]. Examples of software-intensive systems are large-scale heterogeneous production systems or business Web service based applications, as well as applications in space and military areas [Biffl et al., 2019]. The increasing complexity of modern software-intensive systems makes it hard for individual developers to cope with every detail of their structure and functionality. Therefore, for systematic knowledge representation, Meier et al. [2019] introduced a unifying terminology and developed criteria for classifying approaches that allow to construct *Single Underlying Models* which have the capability to combine different views on a system, such as requirements, modeling and programming languages. However, this approach does not take into consideration different engineering disciplines which contribute to development of software-intensive systems.

Multi-disciplinary engineering can be defined as a combination of different disciplines such as for example mechanical, electrical or software engineering involved in an engineering process [Biffl et al., 2017a]. Single disciplines often themselves consist of other disciplines. Software engineering discipline for example depends on three main parts: network, software and hardware engineering [Schneidewind, 2012] and different disciplines, such as electrical (elements of hardware) and software (code). Each discipline requires domain specific experts which further use various tools, methods and data models to achieve specific goals of the respective engineering phases.

Knowledge representation is one of the major challenges in MDE environments. In order to build a final artefact, intermediate results created in one discipline are handed over to another discipline as long as the final result isn't achieved. This approach creates a technical dependency between disciplines [Jäger et al., 2011]. Furthermore, not all of the necessary information on the actual engineering objects is recorded or documented. The information which is documented is often very high-level and the documentation is not easily understandable for domain expert from different disciplines. Moreover, the same terms might have different meanings in different engineering disciplines. To overcome some of the in [Jäger et al., 2011] mentioned issues, the authors propose an approach that deviates from the engineering documents for the involved stakeholders with their engineering processes and tasks and finally leads to a cause and effect diagram.

Due to the lack of adequate knowledge representation, testing of system requirements in MDE remains a big challenge. Any system related MDE has to fulfill requirements regarding system functionality, final product quality, functional safety, and information security. After failure or attack systems can be disastrous due to interaction and propagation of failures to the multiple system units. Accidents and failures often happen due to insufficient understanding of failure mechanisms and lack of effective testing and validation [Li and Kang, 2015, Bertolino, 2007].

Systems and software Quality Requirements and Evaluation (SQuaRE) is a series of International Standards (25000-25099) edited by the ISO/IEC organisation and related to Systems and Software Quality. SQuaRE provides, among the others, the framework for quality requirements for systems, software products and data [ISO25030, 2019] and evaluation of software product quality [ISO25040, 2011]. Software product quality requirements are needed for specification, planning, development and evaluation of the system. ISO25030 [2019] provides the list of requirements (Table 2.1) which should be met to assure quality of a system/product.

The framework is developed for various stakeholders, including testers, to verify and validate that the system products meets the expected quality, and project managers, to plan, monitor and control the achievement of the expected quality.

2.1.1 Cyber-physical Production System Engineering

This section explains basic CPPS Engineering and PPR concepts. Section concerning CPPS includes the information necessary for understanding the concept of CPPS in

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

System Requirements	Detailed Characteristics
Functional Suitability	Functional Completeness Functional Correctness Functional Appropriateness
Performance Efficiency	Time-behaviour Resources Utilisation Capacity
Compatibility	Co-existence Interoperability
Usability	Appropriateness Recognisability Learnability Operability User Error Protection User Interface Aesthetics Accessibility
Reliability	Maturity Availability Fault Tolerance Recoverability
Security	Confidentiality Integrity Non-repudiation Accountability Authenticity
Maintainability	Modulatiry Reusability Analysability Modifiability Testability
Portability	Adaptibility Installability Replaceability

Table 2.1: ISO/IEC SQuaRE - system requirements. Based on [ISO25030, 2019].

Industry 4.0, as well as in IIoT. Following it depicts the challenges and requirements for such systems and solution attempts to cover them. Section 2.1.1 includes the description of the product process-resources-concepts, the dependencies and relationships among them, as well, as examples of domains where the approach can be applied.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CPPS concepts

The demand for flexible industrial production (Industry 4.0) requires CPPS. Such systems should be able adapt to shifts in production volume and product variants [Kagermann et al., 2013]. In CPPS the operations of various physical entities are usually controlled by computing cores. Therefore the design of these systems involves several engineering disciplines, such as mechanical, electrical, and software engineering, where experts from different disciplines and multi-disciplinary teams need to be involved. The experts design and use heterogeneous models and tools in order to achieve their objectives. This fact leads to difficulties in knowledge sharing and representation for all stakeholders. One of the main challenges is to make the variety of experts coordinate towards the satisfaction of these requirements, in spite of the fact that they are accustomed to focusing on their domain-specific issues, with their specialized tools. Therefore CPPS Engineering requires integrating models across engineering disciplines [Chabot et al., 2016, Bifflet al., 2021].

Antão et al. [2018] propose Industrial Internet of Things (IIoT) architecture, supported by implementation of the CPPS, which is displayed in Figure 2.1. The architecture consist of five layers. The lowest, perception layer, consists of physical objects, sensors and actuators, having the purpose of sensor data collection and command actuation. The data collected from sensors are further transmitted to the next, middleware layer, using network layer. In the middleware layer information can be stored and analyzed. The processed information is presented to an end user using in the application layer. System administrators can manage and control the overall functionality of the platform in the business layer.

Defining CPPS requirements is a challenging task due to the existing strong dependencies between cyber and physical parts of the system [Zheng and Julien, 2015]. Antão et al. [2018] propose a list of system requirements for validating and testing CPPS in IIoT implementations, as well as identify the main problems regarding test execution. Figure 2.2 represents the defined system requirements. In order to verify CPPS requirements, Chabot et al. [2016] suggests a Requirement Driven Testing Method for Multi-disciplinary System Design, a generic model for test infrastructures and a method to derive specific infrastructures. The authors designed a unified testing framework that allows simulation to play all the test scenarios for a given product. The solution proposes a common system test interface, to be used by experts from every discipline. The starting point of the method is the requirements context. The method suggests two categories of interconnection points with the system, such as the ones which correspond to the normal functional interface of the system and the ones that are needed to run the tests but do not appear in the final product. The method consists of several steps. Firstly, for generic test infrastructure, a requirements engineer inspects specific requirements and creates corresponding instances of this generic infrastructure. The specific test frameworks are system independent and produced only once for all testing activities and product ranges. Following, system designers design models for their specific disciplines from the uniform system interface. From the abstract models, the system architect, together with the system designers, derive an executable/simulatable model. In order to see whether the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

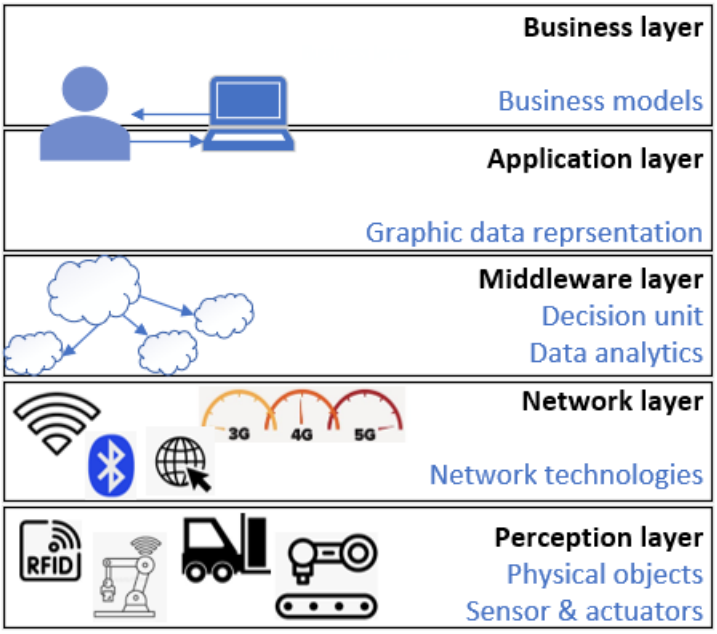

Figure 2.1: Five layer IoT architecture. Based on [Antão et al., 2018].

specific test case passes or fails, the specified behaviour of the testers is translated into an executable scenario automatically. At the final step a validation engineer integrates the new model in the test infrastructure and runs simulation. If the product quality does not meet the requirements, an iterative process is possible. The method focuses on purely multi-physics sub-systems (including components with mechanical, electrical, thermal, electromagnetic physical features, but does not consider hardware/software controlling elements [Chabot et al., 2016].

Li and Kang [2015] propose a strategy for reliability testing and evaluation of cyber physical systems. The strategy includes the testing and evaluation of hardware, software, and architecture reliability, as well as the performance reliability including service reliability, cyber security reliability, resilience and elasticity reliability and vulnerability reliability. The objective of the technology framework are internal factors such as hardware components, software components and architecture. The conditions of the framework are external factors, i.e. environmental conditions, operating and external attacks. The testing and evaluation consist of three parts: systematic analysis, reliability testing and evaluation. In the first part external and internals factors which can influence reliability of the system need to be analyzed. In the second part component reliability and performance reliability are tested. This includes reliability testing of hardware/software components and the network architecture. The performance indices of cyber physical systems are represented by the service, cyber security, resilience and elasticity, and vulnerability. In the presented framework, the four indices are synthesized and the performance reliability

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CPPS req	Testing	IoT layer	Challenges
Scalability	1. Increase number of network nodes	Perception	Latency
	2. Increase available data	Perception	Cost of acquiring/upgrade devices
	3. Increase service availability	Middleware	Constrained data processing methods
Reliability	1. Long term execution	End-to-end	Relationship between anomaly and generated failures
	2. Anomaly injection to generate failures	End-to-end	
	3. Extreme environment conditions	Perception	Code verification methods
	4. Overall counting of received/sent packages	Network	
Security & Privacy	1. Cyber attack injection	End-to-end	Unavailability to inject zero-day attacks
	2. Stealing sensitive data	End-to-end	Simulate known attacks
	3. Security resources are up and running	End-to-end	Lack of expertise in cyber security methods
Timing & Determinism	1. Guarantee cycle time	End-to-end	Identify which component introduces delay
	2. Test equipment process with varying parameters	Perception	Evaluate environmental impact over the process
Safety	1. Simulation of safety process parameters	Perception	Knowing the accident's cause Reliable parameter simulation
	2. Counting number of accidents in the shop-floor	Perception	Availability of relevant environment to test
Recovery	1. Evaluate continuous operation of the system when some of its parts are shut-down	End-to-end	Identify the damage level that precludes system's recovery
	2. Maintain previous state after rebooting	End-to-end	Identify what is the previous state of the system
	3. Analyze time of reboot	End-to-end	Identify the acceptable time of
Interoperability	1. Send messages with non matching semantics or undefined ontology	End-to-end	Communication API does not exists
	2. Integration with 3rd party platforms	End-to-end	Non compatibility between existing APIs
Reconfigurability	1. Analyze time of reconfiguration	End-to-end	Verify reconfiguration in complex systems
	2. Verify system reconfiguration on new topology	End-to-end	Identify acceptable time of reconfiguration
	3. Verify system reconfiguration when a node is added	End-to-end	

Figure 2.2: CPPS testing requirements. Based on [Antão et al., 2018].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

is proposed as the index. This strategy provides support for testing and evaluation of CPS, however, is designed for reliability testing only and does not consider different engineering disciplines such as, for example, mechanical and electrical.

Product-Process-Resource (CPPS) Concept

The main reason of existing of most of the companies is their products, which means that during the value adding process products are created in order to sell them and make profit [Stark, 2015]. Products can be tangible and robust, like, for example a car or intangible like car repair services. CPPSs support production of products by combining suitable production factors. Examples of production factors are raw materials, work-in-progress pieces and complete production resources. A work-in-progress piece can be, for example, a handlebar which has to be transported to another processing station, in order to be finalized. Production resources are all involved in a production process machines, such as welding robots, conveyor belts for transport processes, as well as cameras and physical measurement devices for quality control [Elmaraghy, 2009].

Sauer et al. [2015] introduced Product-Process-Resource (PPR) concept for the dependencies of products produced on production systems based on the production processes. Each product has product characterizing parameters, product components (Bill of Material) and production processes (Bill of Operation) related to it. Production resources execute production processes with their own production characterizing parameters. Each product has all relevant processes required for its manufacturing in its Bill of Operation. Capabilities of the involved production resources defined production process. A set of products is processed by production resources. In order to produce a product, processes are used, that are, in their turn, executed on a resource. This forms dependencies as displayed in Figure 2.3. A graph represents PPR concepts in the following way: the nodes of the graph represent products, processes and resources, while the edges describe dependencies and relationships between them [Sauer et al., 2015].

The PPR concept helps product engineers to describe the manufacturing of products with an executable process [Hundt and Lüder, 2012]. However, it could be used in other domains as well. In agent-based systems, PPR is of special interest [Zavisa et al., 2018], as well as in complexity management, where the complexity of product variants is a topic for investigation [Pfrommer et al., 2015].

PPR Asset Networks (PAN) Concept

This section is based on [Winkler et al., 2021]. In industry, product, process, and resources are often considered individually without expressing their dependencies explicitly. Single assets can contribute to efficient collaboration within CPPS engineering projects in a negative way, that, in case of overseen dependencies, can lead to additional risks. Therefore the authors propose improving of CPPS Engineering knowledge representation by introducing Product-Process-Resource Asset Networks. PANs are capable of providing views on assets from different viewpoints and enable the possibility of efficient added

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

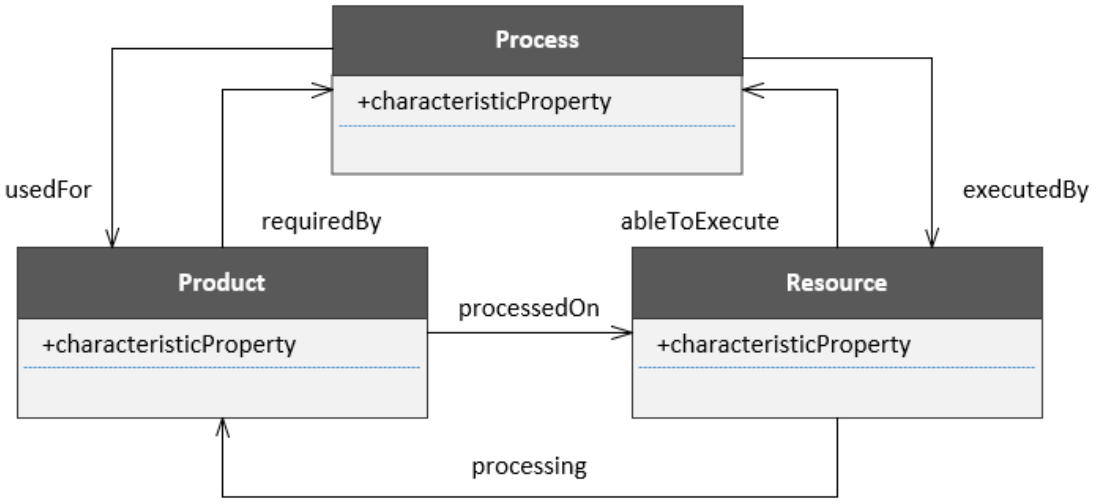

Figure 2.3: PPR concept model. Based on [Sauer et al., 2015].

value application of risks, requirements, and configuration management. Figure 2.4 displays the PAN meta-model, which includes four building blocks: (1) PPR I4.0 Assets; (2) related Asset Properties; (3) PPR Network Nodes; and (4) PPR Network Links. PPR assets can be a part of another assets and have asset properties, which could contain other properties. PPR assets are PPR network nodes which can be linked with other nodes by PPR network links. Network links can be of an incoming or outgoing type, while the nodes are either sources or targets for the links.

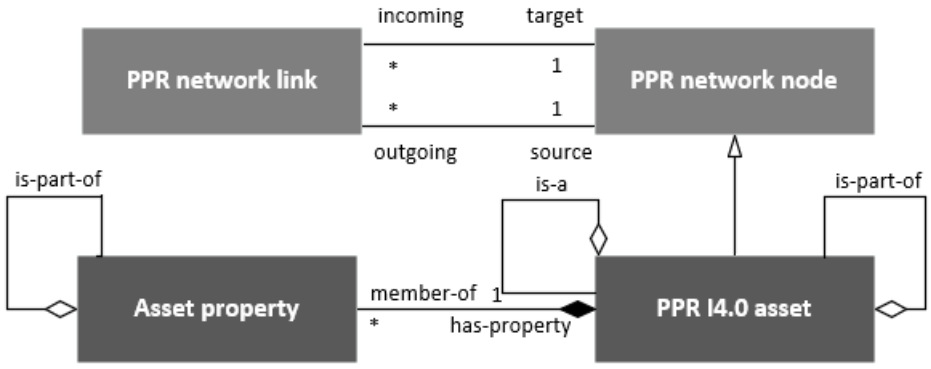

Figure 2.4: PAN concept meta-model. Based on [Winkler et al., 2021].

Figure 2.5 provides visualisation of a derived from the meta-model PAN for Industry 4.0 Testbed located at CTU in Prague [1]. The use case concerns the production line which consists of a set of basic production operations: pick a component from a predefined place by robot; place a component to a predefined place by robot; move a semi-product on a

[1]Industry 4.0 Testbed: `ciirc.cvut.cz/teams-labs/testbed/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

shuttle. Operations pick and place are merged into Pick&Place due to the time efficiency. In order to implement a PAN, stakeholders first provide knowledge of individual aspects of engineering assets from different points of view. This step includes identification of assets and links between them, network construction, verification and validation, and feedback and improvements. Once the mentioned above knowledge has been collected, the PAN is built. On top of the PAN selected added value components are added, such as risk assessment and mitigation during Engineering and operation, requirements tracing and CPPS configuration. In this thesis the domain knowledge models are represented using PAN.

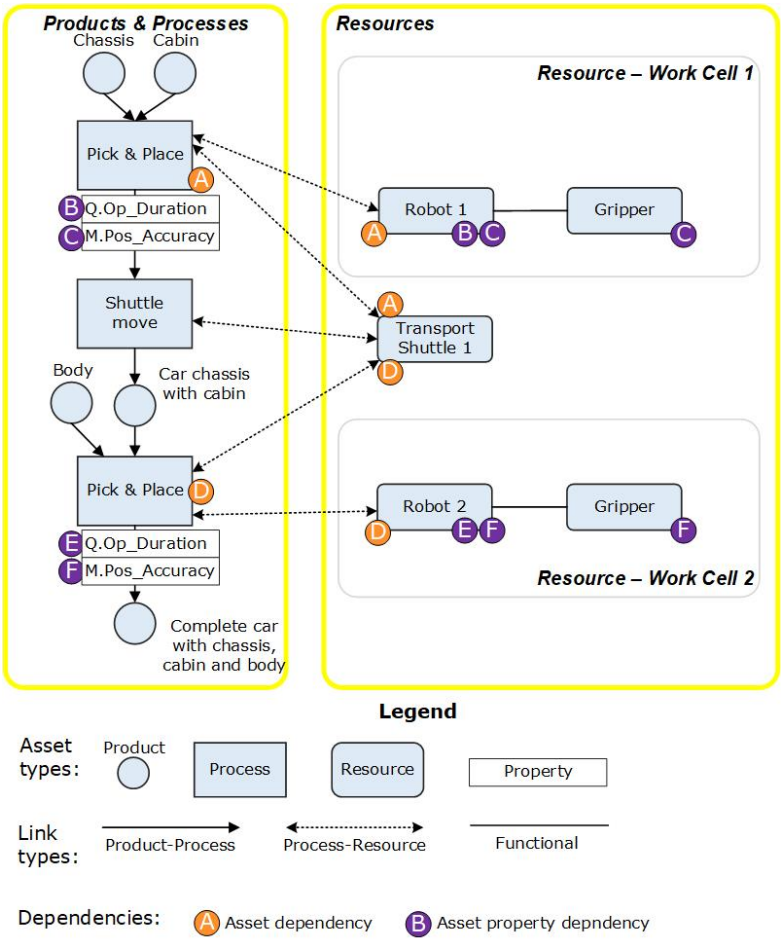

Figure 2.5: PAN of selected assets: robots connected by a transport shuttle. Based on [Winkler et al., 2021].

2.1.2 Software System Engineering Concepts

The goal of this sections is to define and describe the basic concepts related to software system engineering, as well as to identify the system requirement and the challenges that

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

are present in the process of software testing.

Humphrey [1988] suggests the following definitions for software, software engineering, software quality and software process:

- *Software* is a program and all of the associated information and materials needed to support its installation, operation, repair and enhancement.
- *Software engineering* is the disciplined application of engineering, scientific, and mathematical principles and methods to the economical production of quality software.
- *Software quality* is the degree to which a product meets its users' needs, This may refer to functional content, error rates, performance, extendibility, usability, or any other product characteristics which are important to the users.
- *Software engineering process* is is the total set of software engineering activities needed to transform a user's requirements into software. This process may include requirements specification, design, implementation, verification, installation, operational support, and documentation. It also may include either temporary or long term repair and/or enhancement to meet continuing needs. The process may require that the professionals learn the users' needs, translate them into a requirements definition, and modify this definition while gaining more knowledge during design and implementation.

In addition to the quality metrics mentioned in the above stated definitions, Felderer and Schieferdecker [2014] define and explain software system requirement depicted in Table 2.2. Verification of the system requirements require different testing test methods, i.e., functional testing, security testing, and performance testing. The system requirements, so-called, risk drivers determine which testing method is appropriate and should be chosen.

System Requirements	Description
Reliability	The system is able to deliver services as specified
Availability	The system is able to deliver services when requested
Safety	The system is able to operate without harmful states
Security	The system is able to remain protected against attacks
Resiliency	The system is able to recover timely from unexpected events

Table 2.2: Software system requirements. Based on [Felderer and Schieferdecker, 2014].

2.1.3 Summary

Knowledge sharing and testing are both challenging in MDE. While in software engineering the main challenges are efficiency, test coverage and human factors, in CPPS it is, in

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

addition to the challenges mentioned above, executability, system recovery and fragility issues. The challenges in CPPS add additional risks not only to the system itself, but also to the quality assurance process. Such risks need to be identified as well. Risk assessment is one of the techniques which allows experts to identify risks Biffl et al. [2021]. According to Felderer and Schieferdecker [2014], system requirements are risk drivers which require different testing approaches, such as, for example, functional testing and performance testing. Such testing techniques allow not only risks mitigation, but also risk avoidance and system improvements.

2.2 Risk Management

"If you always do what you always did, you will always get what you always got" [Stamatis, 2019]. In other words, in order to get a different result, a change in the process or actions is needed. Changes always include potential risks, which should be identified, assessed and prioritized accordingly. This section summarizes related work on risk management and risk assessment in CPPS engineering. Risk management is essential in every engineering process [Hopkin, 2018]. Identifying and mitigating project related risks are necessary steps for managing projects successfully [Carbone and Tippett, 2015]. Risk assessment determines the significance of the risks and builds the basis for the risks mitigation to prevent defects and avoid the recurrence of defects [Felderer and Ramler, 2014, Hopkin, 2018].

2.2.1 FMEA - Failure Mode and Effects Analysis

This subsection aims to discuss failure mode and effects analysis (FMEA) and is mostly based on [Stamatis, 2019]. FMEA is an engineering risk management method which helps the stakeholders to define, identify, prioritize and eliminate possible system, design or process failures before they reach the end customer. The goal of the method is elimination of failure modes or reducing the risk of their occurrence. FMEA as well provides a framework for critique of a system design or a process. By following the method, the experts can prevent problems, rather than react to them.

FMEA is a beneficial method for various engineering industries and, if properly conducted, it can add the following value to the risk management process:

- confidence that all reasonable risks have been identified early in the process and the appropriate counterarguments have been taken or defined;
- priorities for a product or process improvements have been set;
- rework, scrap and manufacturing costs are reduced;
- knowledge of the product and the processes is stored;
- productions failures and warranty costs are reduced;

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- risk and actions are properly documented for future re-use.

In order to conduct FMEA effectively, the eight-step method presented in Figure '2.6 has to be followed.

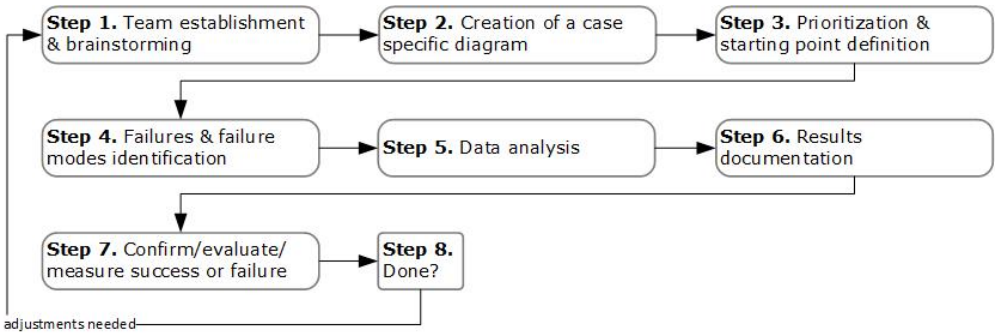

Figure 2.6: The 8-steps FMEA process. Based on [Stamatis, 2019].

- **Step 1**. Establish a team which includes appropriate team members and do brainstorm. The team must be cross-functional and multi-disciplined. The participants must be willing to participate. The brainstorming part includes the definition of areas for improvements, such as system, design, process, product or service, and the opportunities for improvements. The brainstorm should be properly documented using affinity diagram, storybook method and cause-effect diagrams.
- **Step 2**. Create a functional block diagram for FMEA of a system or design or a process, or a service flowchart for FMEA of a process. This helps to make sure that all the participants of the FMEA understand the objectives of the unit the FMEA is conducted for, and relevant problems.
- **Step 3**. Prioritize the important parts and define the starting point of the FMEA.
- **Step 4**. Begin data collection of the failures and categorize them. Identified failures are failure modes of FMEA.
- **Step 5**. Analyze the data by retrieving information from it, applying the expert knowledge and making decisions.
- **Step 6**. Record results in order to determine the severity, occurrence, detection and risk priority number.
- **Step 7**. Confirm/evaluate/measure the success or failure by answering the following question: how has the situation changed: improved, got worse or has not changed? Based on the answers on the above stated question, recommendations for future actions are done.
- **Step 8**. Repeat the steps again, since FMEA is a continuous process.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

2.2.2 Multi-view Risk Assessment in CPPS Engineering

This section is about risk assessment in CPPS engineering mainly based on [Biffl et al., 2021]. *"Risk assessment in CPPS engineering focuses on identifying and analyzing product, process, and resource risks that might lead to defective products caused by inaccurate or defective processes or resources, omitting intentional wrongdoing" [Biffl et al., 2021].* In order for stakeholders to conduct risk assessment, Biffl et al. [2021] propose a CPPS Risk Assessment Meta-Model depicted on Figure 2.7 and a CPPS Risk Assessment method (CPPS-RA) display in Figure 2.8.

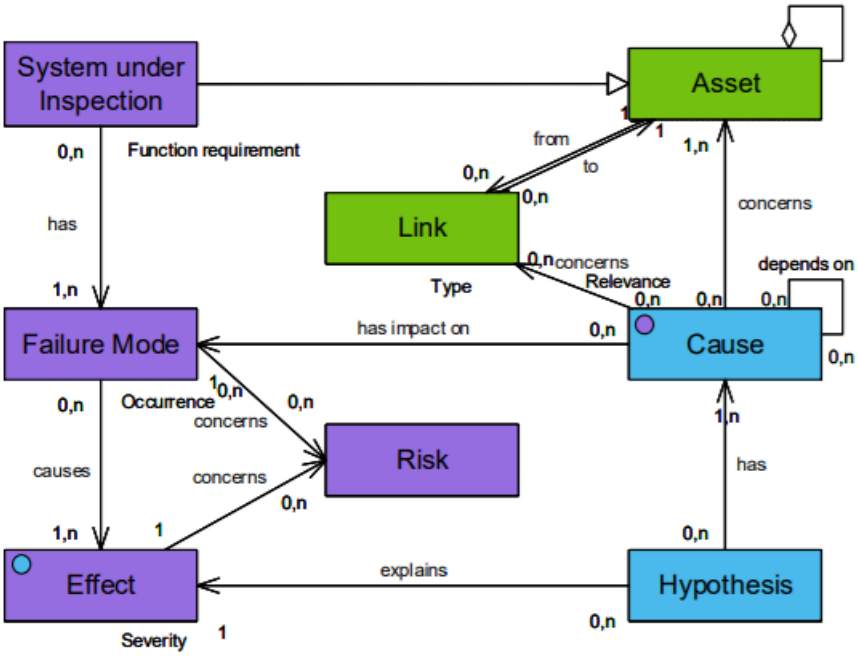

Figure 2.7: CPPS-RA core concepts meta-model [Biffl et al., 2021].

The multi-view RA model combines three groups of concepts represented in separate views: Failure Modes and Effects Analysis (FMEA) concepts (in violet), CPPS Engineering Network (CEN) concepts (in green) and Cause-Effect Hypothesis concepts (in light blue) (Figure 2.7). The multi-view model allows to build multi-view cause-effect diagrams which can further be used for risk assessment. In the standard FMEA process, in order to to assess risks in the System under Inspection (SuI), a multi-disciplinary team of experts conducts the process by applying a set of guidelines and templates [Carbone and Tippett, 2015]. The main steps include: 1) identification of assets, possible issues, and failure modes; 2) elicitation of effects and causes; and the risk assessment. While the FMEA approach is well established single engineering disciplines, for multiple disciplines it requires support of multi-view CPPS Engineering Networks (CEN). CEN is based on PPR concepts explained in 2.1.1. The engineering domain experts identify assets and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

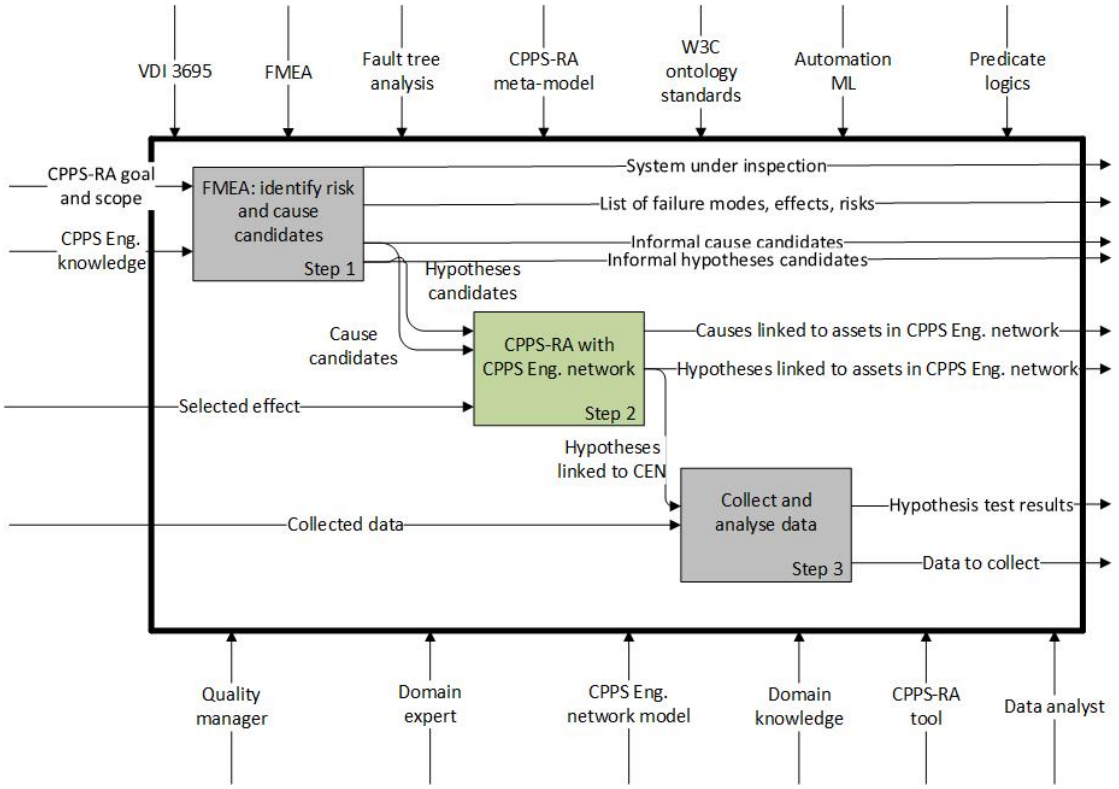

Figure 2.8: CPPS-RA method overview (in IDEF0 notation [Presley and Liles, 1995]) [Biffl et al., 2020b].

link them appropriately [Biffl et al., 2021].

CPPS-RA method contains three main steps and uses the concepts displayed in Table 2.3. The steps are: 1) FMEA: Identify risk and informal cause candidates. The inputs for this step are CPPS-RA goal and scope, as well as CPPS engineering knowledge. Expected outputs are cause candidates and hypothesis candidates; 2) CPPS-RA with CEN exploration. With regards to the output from the previous step, additional inputs for step two are needed, such as informal cause candidates, informal hypothesis candidates and a particular effect. After the step has been conducted, causes linked assets in CEN and hypotheses linked to CEN are identified; 3) collect and analyze data based on CPPS-RA results. Based on the hypotheses linked to CEN, data is collected and analyzed. CEN exploration step further consists of three steps: 1) explore CEN. Selected effect and informal caused candidates are taken into consideration and causes linked to assets are established; 2) analyze cause candidates and cause-effect pathways. Based on the causes linked to assets, cause candidates and pathways are analyzed and causes linked to CEN and to the effect are identified; 3) build hypothesis linked to the CEN. In the last step, hypothesis linked to CEN are built based on the cause candidates from the step two. The steps are displayed in Figure 4.5. [Biffl et al., 2021].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Concept	Concept description
SuI	System under Inspection, an Asset
FM_x	Failure Mode x of the SuI
E_{xx}	Effect xx
A;A_F, A_P, A_{P_0}, A_R	Asset; a Function (F), Product or Material (P), Process (P'), Resource (R) Asset in a CEN
$L_t(A_x, A_y)$	A Link of type t between two Assets, A_x and A_y
$C_x(A_y)$	Cause Cx associated to asset Ay, e.g.,a wrong parameter value leading to a FM.
$H_x(E_{xx}, C_x)$	Hypothesis, linking effect E_{xx} to a set of causes C_1, ..., C_n via a pathway of assets and links in the CEN

Table 2.3: Key concepts for Risk Assessment with FMEA in a CEN. Based on [Biffl et al., 2021].

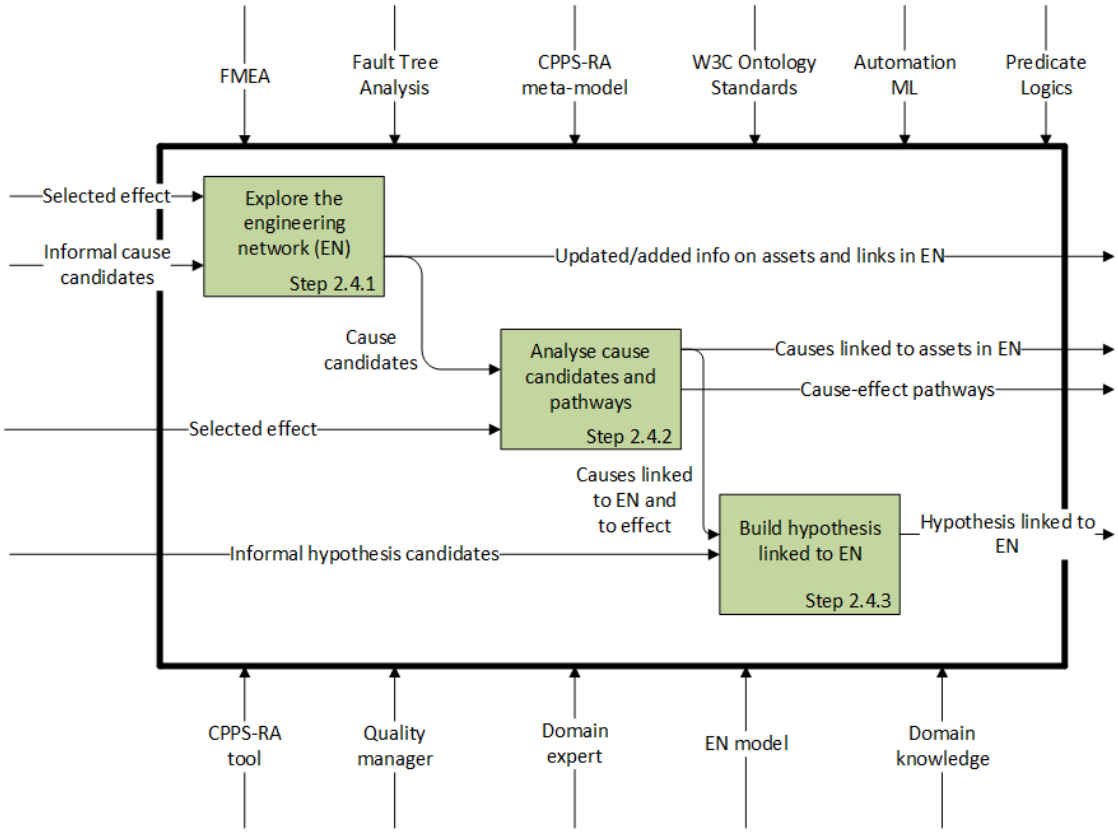

Figure 2.9: CPPS-RA CEN Exploration (in IDEF0 notation [Presley and Liles, 1995]) [Biffl et al., 2021].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.2.3 Summary

This work aims at improving testing related knowledge representation of the system in multi-disciplinary engineering environments by extending the CPPS-RA core concepts meta-model [Biffl et al., 2021] deriving a multi-aspect cause-effect model from it. This model allows support for the domain experts in concentrating their implicit domain knowledge at one place, considering multiple aspects. Following, risk assessment is conducted with regards to the CPPS-RA method. The output of the risk assessment phase serves as the input for the test case derivation phase.

2.3 Software Systems Testing

Software testing is a method to explore intended effects, such as whether a product matches expected requirements, a production process meets performance goals, as well as to check for risks, which could cause undesired effects. Test process can start at the requirements definition phase and continue throughout the whole Software Development Life Cycle (SDLC) [Tuteja et al., 2012]. This chapter introduces software testing approaches and techniques which are applicable in order to solve challenges mentioned in Chapter 2.1, covering test automation concepts. Following, research gaps are identified and discussed.

Bertolino [2007] suggests a road map of the software testing research that includes most relevant software testing achievements, challenges and dreams. The ultimate software testing dreams, according to the author, are universal test theory, test-based modeling, 100% automatic testing and efficacy-maximized test engineering. However, dreams stay dreams due to the unresolved challenges on the way to achieving them.

- **Universal test theory.** The first dream would be to have a test machinery which ties a statement of the goal for testing with the most effective techniques, to adopt, along with the required underlying assumptions. This dream meets the following challenges: explicit test hypotheses, test effectiveness, compositional testing and empirical body of evidence. The concept of a test hypothesis is required for selection of finite test sets, by which single samples are taken as the representative of several possible executions. Test hypotheses are often implicit, while it would be extremely important to make them explicit. Test effectiveness is effectiveness of existing and new test criteria. To understand the classes of faults for which the criteria are useful, analytical, statistical, or empirical evidence of the effectiveness of the test-selection criteria in revealing faults should be provided. Compositional testing means understanding of how the test results observed in the separate testing of the individual piece can be reused. Empirical body of evidence is a challenge which includes the following needs: controlled experiments to demonstrate techniques, collecting and making publicly available sets of experimental subjects, as well as industrial experimentation.
- **Test-based modeling.** The second dream refers to developers being fully aware of testing challenges. In this case they would provide models with built-in testing

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

knowledge. This dream meets the following challenges: model-based testing, anti-model-based testing and test oracles. Even though model-based testing has been researched since more than half of of the century, it is not widely used in the industry. Similar to model-based testing, anti-model-based testing, which refers to developing models after the testing phase, is in on the lower side of usage by the industry. Test oracles are the methods that provide the expected outputs for each given test case. Determining such methods is a great challenge.

- **100% automatic testing.** The dream would be a powerful integrated test environment which by itself, as a piece of software is completed and deployed, can automatically take care of possibly instrumenting it and generating or recovering the needed scaffolding code (drivers, stubs, simulators), generating the most suitable test cases, executing them and finally issuing a test report. This dream meets the following challenges: test input generation, domain-specific test approaches and on-line testing. Test input generation is a critical challenge since it is not always easy to determine what data can be generated automatically, what data should be random vs fixed, and how the data should be generated. Domain-specific test approaches challenge refers to the kind of application being observed. On-line testing concerns mainly the WHERE and WHEN to observe the test executions.

- **Efficacy-maximized test engineering.** All theoretical, technical and organization issues surveyed so far should be reconciled into a viable test process yielding the maximum efficiency and effectiveness (the two words together make it efficacy). This dream meets the following challenges: controlling evolution, leveraging user population and resources, testing patterns, understanding the costs of testing and education of software testers. Controlling evolution challenge is about what, where and when to replay executions following software evolution. Leveraging user population and resources challenges means understanding of how users challenge the time and places of software runs. Testing patterns challenge concerns how to test and how much to test. Education of software testers challenge is about the coverage of all characterizing aspects of testing, which is often not the case.

2.3.1 Test-driven development

Test-driven development (TDD), also called test-driven design discussed in [Beck, 2002] is a software development approach relying on software requirements being represented as test cases prior code implementation, and tracking software development process by testing the software under development against the specified test cases. In the usual software development approach software developed first and test cases are specified later. [Beck, 2002] proposes the sequence of steps for a TDD cycle depicted in Figure 2.10.

While TDD approach is well established in software engineering domain, it is not well known by CPPS engineers. Gherkin notation can be used as early as in the requirements specification phase [Winkler et al., 2018] and allows domain experts to specify test cases

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

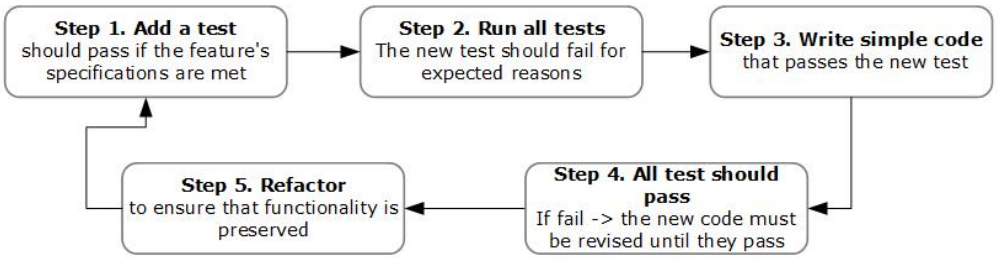

Figure 2.10: Test-driven development cycle. Based on [Beck, 2002]

using Gherkin[2] discussed in [Micallef and Colombo, 2015] in close to human language syntax. Therefore Gherkin test scenarios can be used for TDD [Meixner et al., 2020].

2.3.2 Risk-based testing

Testing of software-intensive systems faces multiple decision problems [Felderer and Ramler, 2014]. First of them is selecting the sets of tests that assure the specified properties of the system and have the ability to identify the defects that harm the critical parts of the system. In addition, due to limited resources and time constraints, testing is often done under high pressure regarding time and effort. Solutions for both of the stated decision problems can be supported by risk-based testing approaches where the identified risky elements of the system from the risk assessment part are the entry point for the Risk-based Test (RBT) strategy [Felderer and Schieferdecker, 2014, Grossmann et al., 2020]. The taxonomy of the RBT is proposed by [Felderer and Schieferdecker, 2014] and displayed in Figure 2.11. The first branch represents system requirements: functionality, safety and security. They are the main drivers of system risks. In order to identify and analyze risks related to the risk drivers, risk assessment method is needed. The methods includes identification of risk item types, such as functional, architectural, development, runtime, test or generic risk artifacts. Following risk factors are identified and analyzed, such as risk exposure, risk likelihood and impact rating. The risks are then estimated and the degree of automation is identified. The lower branch represents risk-based testing process which includes standard test process steps based on the identified risks: test planning, test design, test implementation, test execution and test evaluation.

2.3.3 Software testing techniques

For any system which behavior is controlled by software it is impossible to test all combinations of possible parameters and their impacts. Therefore test case design is an effort-consuming and error-prone process. Based on the cause candidates set from the risk assessment phase, test cases can be derived using the test case design techniques such as equivalence class partitioning, boundary value testing and decision making testing which ensure efficient coverage of risks [Spillner et al., 2014]. However, the designed

[2]Gherkin: `https://cucumber.io/docs/gherkin/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

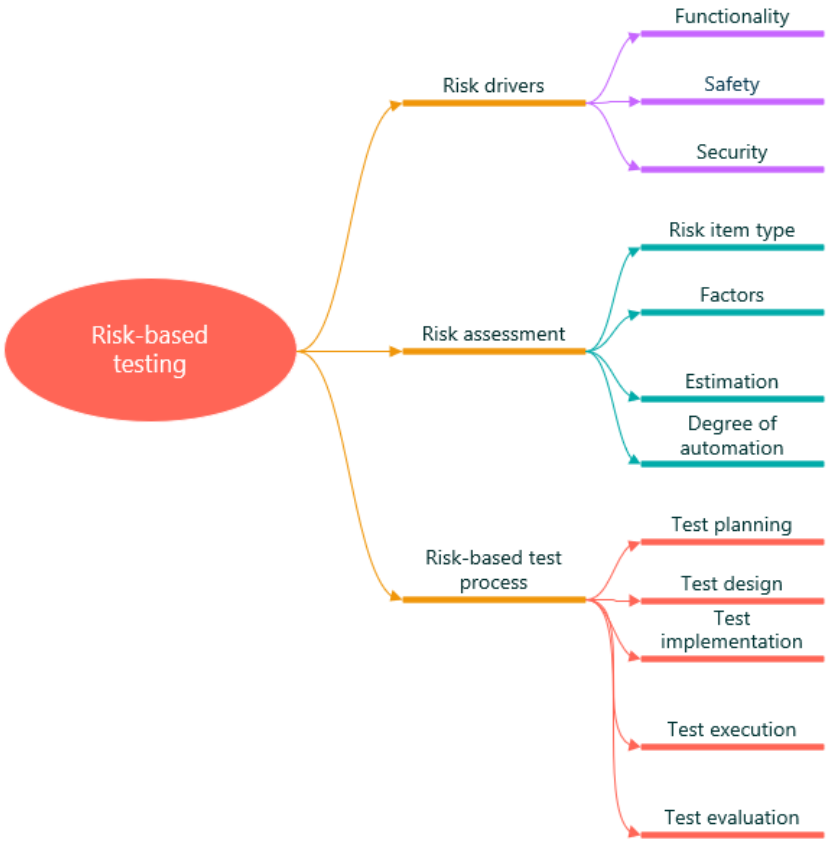

Figure 2.11: Overview of risk-based testing taxonomy. Based on [Felderer and Schieferdecker, 2014]

tests set could be not the most efficient in terms of testing costs and effort. In order for a test engineer to derive the minimum set of test cases to achieve the maximum coverage, the following should be considered: a) not every cause contributes to every effect and most effects are caused by interactions between relatively few causes [Kuhn et al., 2010]; b) the test cases have different impact and some of the test cases may not detect any fault [Ahmed, 2016]. Therefore, the initial test case set should undergo the process of test set size minimization, optimization and prioritization in order to reduce testing effort and costs [Kuhn et al., 2010, Ahmed, 2016, Note Narciso et al., 2014]. This section presents related work in area of Software system testing, which includes testing challenges, goals and metrics, as well as testing strategies and approaches.

Equivalence class partitioning

This section is based on [Burnstein, 2003, Myers, 1979]. Equivalence Class Partitioning (ECP) discussed in [Burnstein, 2003]. is an approach to selecting test inputs for test

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

case design in order to verify software-under-test. Only one representative member of the finite number of equivalence classes, also partitions, is needed to verify the system behaviour for the whole partition. According to the author, equivalence class partitioning technique is beneficial for testers because it eliminates the need for exhaustive testing, allows them to select a subset of test inputs with a high probability of detecting a problem and reducing the test data space. In order to select the right equivalence classes, the following must be considered:

- Both valid and invalid equivalence classes must be considered. Invalid equivalence classes can represent unexpected inputs or error states.
- It might be wise to consider equivalence classes for output conditions.
- The derivation of equivalence classes is a heuristic process, which improves with the growing experience of the tester.
- In some cases, derivation of the equivalence classes might be very hard or impossible for a tester alone. In such cases, help of analysts or other stakeholders is required.

Myers suggests a set of conditions as guidelines for testers in order to select an appropriate set of input equivalence classes [Myers, 1979]:

- *"If an input condition for the software-under-test is specified as a range of values, select one valid equivalence class that covers the allowed range and two invalid equivalence classes, one outside each end of the range."*
- *"If an input condition for the software-under-test is specified as a number of values, then select one valid equivalence class that includes the allowed number of values and two invalid equivalence classes that are outside each end of the allowed number."*
- *"If an input condition for the software-under-test is specified as a set of valid input values, then select one valid equivalence class that contains all the members of the set and one invalid equivalence class for any value outside the set."*
- *"If an input condition for the software-under-test is specified as a "must be" condition, select one valid equivalence class to represent the "must be" condition and one invalid class that does not include the "must be" condition."*
- *"If the input specification or any other information leads to the belief that an element in an equivalence class is not handled in an identical way by the software-under-test, then the class should be further partitioned into smaller equivalence classes."*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Experimentation in Software Engineering

"Experiment (or controlled experiment) in software engineering is an empirical enquiry that manipulates one factor or variable of the studied setting [Wohlin et al., 2014]". This section is based on [Wohlin et al., 2014]. Validity of experiments design and their outputs is achieved by appropriate definition of data, objects and subjects, setup and constraints. Humans apply different treatments to objects in human-oriented experiments, while in technology-oriented experiments, different technical treatments are applied to different objects. Unlike case studies, experiments require high levels of control over the setup. The objective of this method is to change treatments of one variable (cause) at once, while the other variable remain fixed, and observe the correlation to the dependent variable (effect). An Experiment usually consists of the following steps: 1) Scoping; 2) Planning: selections of the context, formulation of hypotheses, selection of variables and subjects, experiment design, instrumentation, validity evaluation, description of validity threats and their priorities; 3) Operation: preparation, execution and data validation; 4) Analysis and Interpretation; 5) Presentation and Package.

Thanks to the high level of control of the experimental setup, precise data analysis and a high level of reproducibility can be achieved. However, with the level of control costs of the research methods might raise.

2.3.4 Test automation

When designing a software automation method, the following should be considered: decision making process on what can be automated and what makes sense to automate, selection of appropriate tools and notations which support the system under test requirements and tests representation needs, continuous integration tools selection, evaluation methods, as well as result data representations [Winkler et al., 2018]. An efficient test case generation approach based on product and process properties and preconditions was discussed in [Meixner et al., 2020]. Once the test cases have been generated, based on the multiple conditions, such as whether it is possible and makes sense to automate them, the sub-set of the test cases can be automated. One of the possible representation of test cases to automate, Gherkin notation discussed in [Micallef and Colombo, 2015]. Gherkin allows experts with and without coding skills derive parameterized test cases which can be automated by experts with coding skills. In order to implement Gherkin test cases, the domain experts agree on a vocabulary and use only this vocabulary to ensure the common understanding of the test cases and efficient test case automation. Another suitable, keyword-driven [Rwemalika et al., 2019] approach requires better technical understanding from the experts implementing the test cases. In this approach every method or function is represented by a keyword. This allows to achieve very high granularity and reusability of the keywords and yet a human-readable code [Hametner et al., 2012]. Robot framework is a commonly used tool for test automation and robotic process automation which supports the keyword-driven approach [Bisht, 2013] and allows introduction of Gherkin layer on top of the keywords. Implemented test cases can undergo

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

parameterized execution based on a defined schedule in scope of continuous integration with support of software tools, such as for instance Jenkins[3] tool [Duvall et al., 2007].

2.3.5 Summary

In order to derive the minimum amount of test cases to fulfill the domain related testing needs, in this research, based on the identified risks, equivalence class partitioning technique in combination with test selection strategies is used. In order to be able to set reference test values for a step of the multi-aspect test case specification and automation based on cause-effect analysis method, an experimentation approach is needed. In this thesis experiments are integrated into the case studies in order to allow domain experts to define reference values which are missing in system specifications. Knowledge of the reference property values of the system support defect detection and system optimization activities. Following, a sub-set of the derived test cases could be automated using established notations and tools. This research aims to application of the well established software engineering approaches, methods and tools in CPPS engineering context.

[3] Jenkins: https://www.jenkins.io/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

Research Issues and Approach

Chapter 1 introduced the testing challenges in software-intensive domain as motivation for this thesis. Multi-aspect cause-effect relationships were explained in context of the domain. Chapter 2 provided background information and recent research regarding the investigated topics in various engineering environments. In order to overcome the earlier identified testing challenges, the domain specific differences have to be analysed and specified, in order to apply feasible solution approaches from these fields. In addition, analysis of contexts similarities is necessary in order to create an artifact which is generic enough to cover the challenges across different contexts.

Section 3.1 presents the main Research Issues (RI) related to *Multi-Aspect Risk Drivers* and *Multi-Aspect Test Case Specification*, and research questions. Further, Section 3.2 discusses the research methodology applied in this thesis, specifically *the design science cycle*, *literature review*, and *qualitative case study research*. Finally, Section 3.2.4 describes in details the specific research approach taken in this thesis and expected outcomes for the research questions.

3.1 Research Issues

This section introduces the main research issues and concerns risk drivers in context of software-intensive systems, multi-aspect cause-effect relationships representation, test case specification and data collection for validation of cause-effect hypotheses, as well as scope and limitations for test case automation. Based on the RIs, Research Questions (RQ) are formulated.

3.1.1 RI1: Risk drivers for multi-aspect testing of system requirements.

According to Systems and software Quality Requirements and Evaluation (SQuaRE)[1], standard [ISO25030, 2019], common software quality requirements include functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability. However, not all of the above mentioned risk drivers motivate multi-aspect risk-based testing. Furthermore, in context of the thesis, different types of systems such as software engineering and cyber-physical production systems are investigated. Therefore it is important to identify sets of risk drivers for each of the contexts. The context motivates the first research question:

RQ1: *Which risk drivers motivate multi-aspect risk-based testing of cause-effect hypotheses between production process characteristics and production resource parameters/influences?*

The outcome of this research question are sets of multi-aspect risk drivers representing quality requirements for subsets of software-intensive (production) systems. *Quality managers* benefit from this outcome, as there is a way to categorize risks and encapsulate the categorization into the risk-based testing process. *Domain experts* and *quality engineers* benefit from the outcome as well due to the fact that they could use the categories of risks in order to enable better communication and involve the domain experts with the appropriate knowledge. Furthermore, according to the risk driver, different testing activities can be executed separately.

Findings of the risk drivers definition are presented in workshops to stakeholders of the domain experts to discuss them in regards to real challenges and experiences from daily business. Expected feedback is incorporated and a set of main risk drivers concluded. Based on the identified risk drivers, use cases for evaluation are elicited.

3.1.2 RI2: Representation of multi-aspect cause-effect hypotheses.

In order for a *domain expert* to build cause-effect hypotheses and share knowledge to support validation of the hypotheses, the following requirements have been derived from the literature [Biffl et al., 2017a, Meixner et al., 2020] and discussions with the industry company partners, based on the described in Section 1.1.2 challenges: **Requirement 1.1:** Representation of domain specific and testing concepts. **Requirement 1.2:** Capability to store test related information for each asset property. **Requirement 1.3:** Capability to query the model in order to extend the knowledge base or retrieve the information. The context motivates the second research question:

RQ2: *What knowledge model can represent multi-aspect cause-effect hypotheses between production process characteristics and production resource parameters/influences?*

Domain experts require a model to build hypotheses on cause-effect relationships. To address RQ2, *a multi-aspect cause-effect model* from the extended *CPPS-RA core concepts*

[1]SQuaRE: `https://iso25000.com/index.php/en/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

meta-model [Biffl et al., 2021] is derived. The model should allow storing of cause-effect related characteristics on each of the cause assets. This work aims to design the meta-model and the domain knowledge models derived from the meta-model. Both should be evaluated in the context of selected use cases and hypotheses in cooperation with domain experts.

3.1.3 RI3: Multi-aspect test case specification and automation to validate cause-effect relationships.

Quality engineers require a method to define the minimum set of test cases to validate the multi-aspect cause-effect hypotheses. In order for a *test engineer* to derive multi-aspect test cases to validate the cause-effect hypotheses, the test case specification method requires: **Requirement 2.1:** An easy understandable representation of the key concepts of the test case specification approach. **Requirement 2.2:** The test case specification receives the cause-effect hypotheses from the risk assessment step. The risk assessment method needs to allow to provide the test case specification method with prioritized risk and the related cause-effect hypotheses.

RQ3: *What are the process steps to support multi-aspect testing in CPPS and Software engineering?*

A method which covers the above stated requirements should be designed. The goal of the method is to allow quality experts to define an efficient test set. In order add more details to RQ3, detailed RQ3a and RQ3b have been designed: ***RQ3a:*** *What is the minimum set of test cases and test data a test engineer requires to validate a hypothesis based on cause-effect relationships?*

To address RQ3a, a method with a sequence of steps should be designed to define the test planning and execution activities for cause-effect hypotheses validation. The method should cover the following requirements: **Requirement 2.3:** Quality experts should be able to minimize the amount of derived test cases. **Requirement 2.4:** Capability to collect and evaluate reference data and data which lead to the effect from the multi-aspect cause-effect model. Capability to decide whether or not to reject the hypothesis. The goal is to allow test engineers to define a systematic test suite to define a minimal set of tests that collect data on cause-effect relationships to validate the hypotheses. The method for deriving test cases for selected use cases and hypotheses should be evaluated in cooperation with domain experts.

Test automation engineers require a method to define the scope of test case aspects to automate, and limitations. In order for a *test automation engineer* to automate the multi-aspect test cases, the multi-aspect test case automation method should include: **Requirement 2.5:** Exploration of the scope of test case aspects to automate, and limitations. Selection of test case aspects to automate. Selection of test case representation, design of test cases for automation. The context motivates an additional research question:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

RQ3b: *What are the test scope and limitations for multi-aspect test automation?*

To address RQ3b, the multi-aspect test case specification method should be extended with description of steps for test automation scope and limitations. The method should be evaluated by its application for selected use cases in cooperation with domain experts.

3.2 Research Methodology and Approach

The goal of the thesis is to design and evaluate a novel method for multi-aspect testing by adapting established models, methods, and mechanisms from informatics discipline by following the design science approach represented in a Visual Abstract [Engström et al., 2020], including series of case studies and experiments. This section aims to summarize existing research methods which were employed in this master thesis in order to achieve specified goals.

3.2.1 Design Science Research

Design Science Research (DSR) is a methodological framework proposed for design and investigation of artifacts in context, in order to improve the current status of or solve identified challenges. The goal of design science research is to develop knowledge about unresolved problems, in order for professionals to apply the knowledge and solve the problems. Multi-aspect test case specification and automation based on cause-effect analysis has not been researched yet. The research questions of the thesis focus on finding answers to new unresolved questions, such as multi-aspect risk drivers, multi-aspect knowledge representation and test case derivation and automation. Therefore, DSR is an appropriate research framework for this thesis. DSR aims to support researchers to solve technical research problems stated according to the template:

How to <(re)design an artifact>
that satisfies <requirements>
so that <stakeholder goals can be achieved>
in <problem context>? [Wieringa, 2014]

A major advantage of DSR is its flexibility. The methodology allows encapsulating other research methods within its cycle. However, flexibility is also a disadvantage due to the possibility of different interpretations of the guidelines. Therefore, DSR is often mistakenly considered to be a standalone research method.

This thesis is structured and conducted according to the guidelines of [Engström et al., 2020] (see Figure 3.1). The visual abstract template provided by Storey et al. has three major components: *"A) the theory proposed or refined in terms of a technological rule; B) the empirical contribution of the study in terms of one or more instances of a problem-solution pair and the corresponding design and evaluation cycles; and C) the*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

assessment of the value of the produced knowledge in terms of relevance, rigor, and novelty" [Engström et al., 2020].

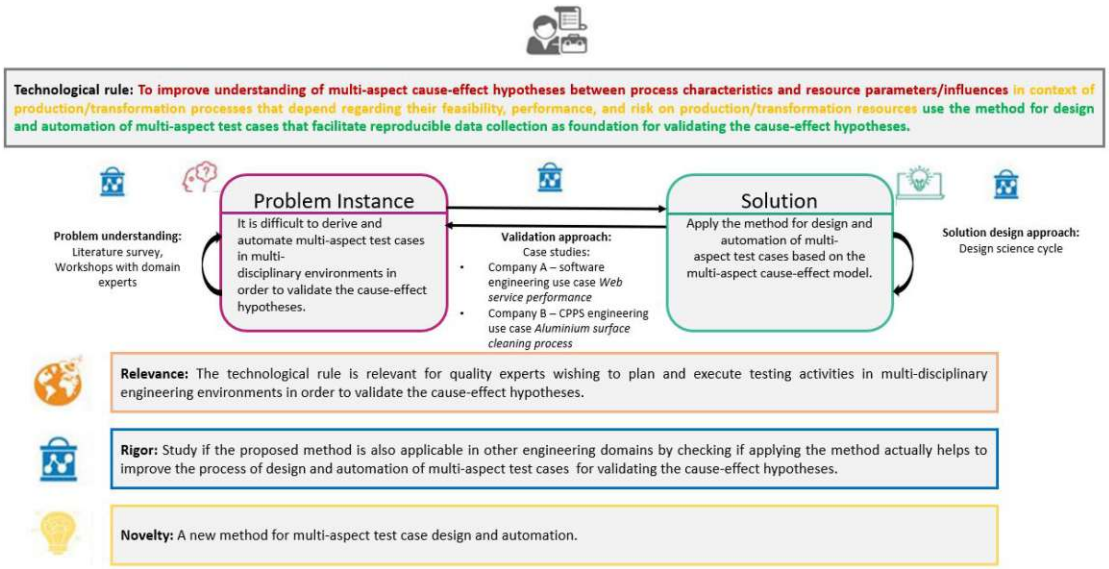

Figure 3.1: Visual abstract identifying the main elements of the proposed research. Based on [Engström et al., 2020].

Technological rule. *To improve understanding* of multi-aspect cause-effect hypotheses between process characteristics and resource parameters/influences *in the context of* production/transformation processes that depend regarding their feasibility, performance, and risk on production/transformation resources, *design and evaluate* a method for designing and automating multi-aspect test cases that facilitate reproducible data collection as foundation for validating the cause-effect hypotheses.

Problem instance. It is difficult to derive and automate multi-aspect test cases in order to validate the cause-effect hypotheses. See the challenges in Section 1.1.

Problem solution. The new method for design and automation of multi-aspect test cases which aim to validate the cause-effect hypotheses.

Problem understanding. The problem related to the multi-disciplinary knowledge implicity [Meier et al., 2020, Biffl et al., 2021], testing challenges [Felderer and Ramler, 2014, Meixner et al., 2020] and test case automation [Winkler et al., 2018, Meixner et al., 2020] related to this matter is well known and reported in several studies.

Solution design approach. Close study of use cases, relevant literature and preliminary workshops with the domain experts who later participated in the evaluation case studies.

Validation approach. Multi-aspect testing experiments with application of the novel approach were conducted in context of Test-driven System Engineering TDSE as case

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

studies with workshops and discussions at:

- Company *A*: use case *Algorithm performance*: use case on performance effects and dependencies that require software-based risk mitigation.
- Company *B*: use case *Aluminium surface cleaning process*: use case on process parameter value improvements that depend on mechanical and chemical dependencies of production resources.

Relevance. The technological rule is relevant for software-intensive systems roles. Quality managers, who have to effectively and efficiently identify multi-domain factors that are likely to lead to specific quality risks, as a foundation for mitigating these risks. Domain experts, who require an efficient method for representing domain knowledge on multi-domain dependencies. Quality engineers, who have to plan and execute testing activities in order to validate multi-domain cause-effect hypotheses. Test automation engineers, who have to support test engineers in automating test cases with multi-domain dependencies.

Rigor. Case studies with domain experts and use cases coming from different software-intensive production system contexts. Study if applying the method actually helps to improve the process of design and automation of multi-aspect test cases for validating the cause-effect hypotheses.

Novelty. In general, single-discipline testing does not consider conditions/dependencies coming from other engineering disciplines or environments. The designed approach goes beyond the state-of-the art [Chabot et al., 2016, Meier et al., 2019, Li and Kang, 2015] by providing a more systematic method for multi-aspect test case specification and automation.

3.2.2 Literature Review

This section concerns The term Systematic (Literature) Review (SLR) and is based on [Kitchenham and Charters, 2007, Biolchini et al., 2005]. SLR refers to a specific methodology of research, developed in order to interpret and evaluate existing literature related to the stated research questions. The goal of SLR is identification of any gaps in current research in order to suggest areas for further investigation. Among the advantages of SLR are higher probability that the results of the literature study are unbiased; evidence on the phenomenon in many contexts, settings and empirical methods; reproducibility. The major disadvantage of the research method is the amount of required effort. Unlike unsystematic reviews, systematic literature reviews follow a well defined and strict sequence of methodological steps defined in the beforehand prepared protocol. However, single steps could be used for every other survey of the related work.

The process of SLR includes three major phases: planning, conducting the review and reporting. The planning phase consists of two key elements: research question formulation

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

and plot establishment. The main phase of the process, conducting the review, is built from the following elements of the SLR: Selection of appropriate search keywords, Search and study collection, Selection of relevant studies, Analysis of primary studies.

Report on the results usually includes abstract and background information, research questions, description of the method, results, discussion and conclusions.

In this thesis only some of the steps of the SLR process are used in order to make the survey more structured and reproducible:

- The preparation phase: a) definition of the research questions; b) initial definition of keywords, c) identification of candidate sources.
- Search and Initial Analysis. This phase included the initial search and basic analysis of the findings. Due to multiple modifications of research questions and extensions of the keywords, this step is an iterative step. Results include the final research questions and a finalized search and analysis process.
- Data Extraction. The results are analyzed according to the research questions. The relevant sources are discussed within the scope of the Chapter 2.

3.2.3 Case Study

Case study (CS) is an empirical research method in social science which provides tools for researchers to study phenomena in their context and collect and generate new knowledge about individual members, groups and entire organisations. CS research method, however, is also a powerful tool for researchers from software engineering and system engineering disciplines. Improvements of software engineering or system engineering processes are complex human-based activities and often differ from organisation to organisation. Success of an engineering project does not only depend on the technical requirements, but also on the psychological aspects. Case study research focuses on answering questions that ask "how" or "why", and where the researcher has little control of present events [Runeson and Höst, 2009, Baxter and Jack, 2010]. Therefore, case studies, which aim to study processes and behaviour of stakeholders in real world settings, are an appropriate research method to study phenomena in system and software engineering contexts. Structure of a CS must include the five main steps [Runeson and Höst, 2009]: Case study design, Planning of data collection, Collecting data, Analysis of collected data, Reporting.

There are different types of case studies: explanatory, exploratory, descriptive, multi-case studies, intrinsic, instrumental, as well as collective CSs. CSs can be designed differently according to their scope. There are three types of CS designs: single study, multi-case study and single case with embedded units study.

Among the major advantages of case study research are: a) the ability to turn researcher's observations into usable data; b) the possibility to use other research methodologies; c) costs and accessibility to users. The major disadvantages include the data analysis effort and inefficiency of the process.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Following text aims to address application of CS research method in this thesis. In order to evaluate the novel testing method, two CSs were conducted using the five steps of the CS research depicted in Figure 3.2.

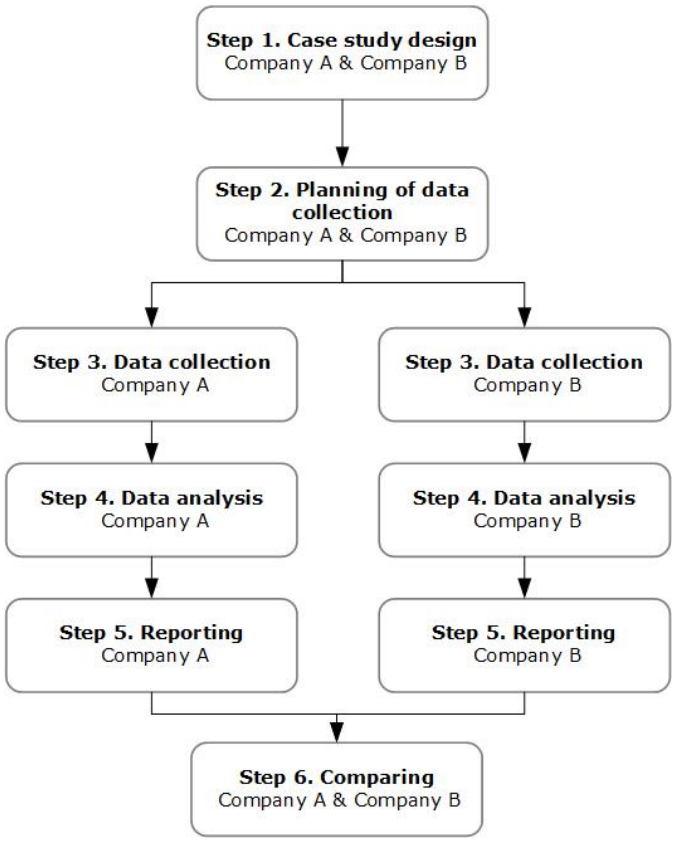

Figure 3.2: Case study design approach for Company A and Company B.

- **Steps 1 and 2: Case study design and Planning of data collection.** The objective of the studies is to evaluate whether the multi-aspect meta-model, the multi-aspect domain-specific knowledge models and the method for multi-aspect test case specification and automation based on cause-effect analysis meet the elicited from the literature review and experience of the domain experts requirements. The studies are considered as two holistic case studies with two units of analysis from two different multi-disciplinary engineering (MDE) domains (company partners): software engineering, Cyber-Physical Production Systems Engineering (CPPS). The hypothesis was formulated: in large-size software-intensive engineering multi-disciplinary organizations it is not easy to test system behavior controlled by software that depends on assumptions, requirements, and designs coming from related "other" systems, engineering disciplines and providers. Therefore, two large-size companies with multi-disciplinary software-intensive system engineering

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

nature were selected in order to evaluate the method in heterogeneous contexts. The participants of the studies are:

Company A (software engineering domain) is the subject of the first case study. Company A is a large-scale company in-house software development. The company employs more than 20.000 employees in 20 countries worldwide and delivers its services to over 20 million registered online customers.

Company B (CPPS domain) a global expert in aluminium manufacturing. The company employs more than 2500 employees in 8 countries worldwide and delivers aluminium products to customers from mobility, construction industry, packaging, electrical and mechanical engineering, medical technology, as well as consumer goods sectors.

For each use case, a case study in cooperation with domain experts in order to evaluate the proposed concepts is conducted. Each case study includes the following phases:

- Introduction phase. During this step the risk drivers table, the knowledge meta-model and model are presented to the domain experts. The domain experts get to know the person who conducts the case study and each other;
- Use case analysis phase. The problem which has to be solved by the designed artefact is discussed and analyzed in details. The analysis includes: contexts of the studies and the problem analysis, involved stakeholders, the traditional approach description;
- PAN building phase. During this phase the domain experts build the PAN network of their system. They depict products, processes, resources and links between them in a knowledge model;
- Derivation of cause-effect hypotheses phase. The domain experts conduct the risk assessment following the risk assessment steps from the test case specification method description and derive the cause-effect hypotheses;
- Derivation of test cases phase. The domain experts derive multi-aspect test case in order to investigate selected effects and associated cause-effect hypotheses;
- Hypotheses validation phase. The domain experts execute the specified test cases and validate or reject the stated in "Derivation of cause-effect hypotheses phase" hypotheses;
- Knowledge extension phase. The results of the tests runs or the assumptions regarding the test cases are stored back to the knowledge graph;
- Automation of selected test cases phase. The domain experts define the scope and limitations of the test case automation based on the method steps. Wherever possible, the domain experts automate the test cases;
- Evaluation phase. The evaluation workshops with stakeholders are designed in order to confirm the rigor of the designed artefacts and collect the opinions of the domain experts regarding its relevance and novelty.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- **Step 3: Data collection.** Based on the requirements collected during the literature review and at the Company A and the Company B during the preliminary workshops, the testing method is designed. The method is evaluated at Company A and the CPPS company partner using workshop approach to find out whether the method improves the process of validating the cause-effect hypotheses. Data is collected during multiple rounds of workshops with multiple domain experts detailing the multi-aspect test case design. Two types of data are collected in order to evaluate the applicability and advantages of the novel method: historical archival and data from the time of the experiment.

- **Step 4: Data analysis.** The data from the conducted workshops is documented textually, allowing further archival analysis of meeting minutes and technical reports. Textual descriptions, business process models and findings discussed with domain experts, who were involved in the data collection and workshop activities, are analyzed. The data collected during the experimental phase is documented and presented in the appendix of this thesis.

- **Step 5: Reporting.** The conceptual evaluation of the method has been published in the 27th peer-reviewed in conference (IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) [Winkler et al., 2022]. The results of the case studies are reported in this thesis (Chapter 5 and Chapter 6).

- **Step 6: Comparing.** The comparative analysis of the evaluation results is presented in Section 7.1.

3.2.4 Research Approach and Results

This section aims to present the research approach used in this thesis to design and evaluate the novel multi-aspect test case specification method. The thesis is motivated by the difficulties in validation of cause-effect hypotheses in software-intensive multi-disciplinary engineering environments. The summary of the research approach is depicted in Figure 3.3. To approach this problem, answers to the three stated research questions have been given. In order to answer RQ1, the literature review to collect the software and CPPS system requirements has been conducted and the relevant sources have been collected. Following, a comparison table has been designed and evaluated in cooperation with multiple domain experts during planned preliminary workshops. In order to answer RQ2, the best practices from the relevant literature have been selected, applied and extended based on the formulated requirements. In order to address RQ2, a meta-model and domain specific models have been designed and evaluated in workshops with domain experts. Then, in order to answer RQ3, RQ3a, RQ3b, the novel method including the usage of artefacts from RQ1 and RQ2 has been designed. The method has been evaluated in two independent case studies in contexts of two different use cases.

Results overview. This part briefly discusses the main results of the thesis in detail. The results are represented by the blue bubbles in Figure 3.3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

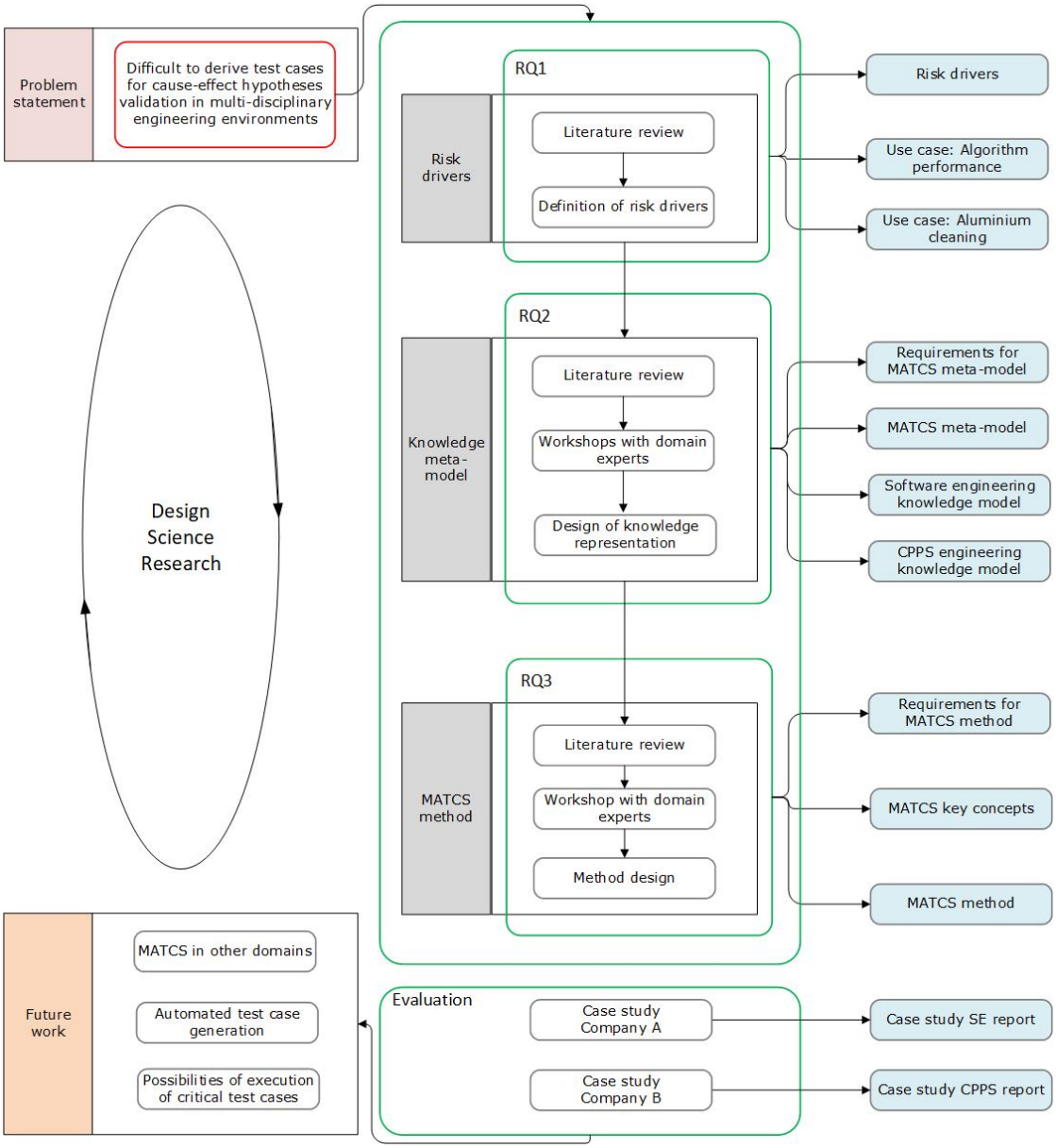

Figure 3.3: Research approach of multi-aspect test case specification method design.

Risk drivers. Based on the literature review a set of system requirements which represent risk drivers has been identified (Section 4.2). The identified set of risk drivers has been discussed with the domain experts in workshops. Based on the selected risk drivers use cases have been elicited.

Use case descriptions. Use cases together with experts from different engineering domains such as software engineering and cyber-physical production system engineering

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

were elicited in order to evaluate the designed artefact (Chapter 5 and Chapter 6).

Detailed requirements definition. Further in this thesis, detailed requirements based on the set of selected use cases and literature review are defined (Section 4.1). The initial set of requirements is presented in Section 3.1.

Identified research gaps. Workshops with domain experts in context of the use cases, requirements and literature survey (Chapter 2) helped in research gaps identification which are presented as challenges in Chapter 1.

Multi-aspect meta-model. The multi-aspect meta model includes view on multiple disciplines and combines different aspects to allow multi-disciplinary cause-effect analysis.

Domain knowledge representation models. One of the outputs of this thesis are a domain knowledge representation models which allow domain experts represent cause-effect relationships (Chapter 5 and Chapter 6). The models are derived from the MATCS meta-model (Figure 4.6).

Test case specification and automation method. The output of this thesis is design of a method that allows addressing test case specification and automation in a range of use cases from different engineering domains (Chapter 4).

Case study reports. To evaluate the feasibility of the proposed method, hypotheses identification, test case derivation, data collection and hypotheses validation, case studies were conducted in cooperation with domain experts. As the output of this activity a case study report for each of the use cases is provided (Chapter 5 and Chapter 6). Furthermore, discussions with the domain experts were held whether the novel method could support TDD in CPPS engineering - Test-driven System Engineering (TDSE) approach, as well as the engineering first approach.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Multi-Aspect Test Case Specification (MATCS) Method

Risk assessment techniques, such as Failure Mode and Effects Analysis (FMEA), contribute very well to the continuous system improvement process. System quality improvements can be considered during various phases of the system engineering life cycle, such as for example design and implementation phases [Juran and Riley, 1999]. Traditional testing methods are not based on risk assessment with FMEA and FMEA does not consider the concept of classes of equivalence. The main objective of this thesis is design of a testing method suitable for multi-disciplinary engineering environments. The method is based on the multi-view risk assessment concept [Biffl et al., 2021] with FMEA where testing drives the risk assessment. Test-driven Development (TDD) is a well known approach in software engineering domain which proposes test case specification or/and implementation prior feature development. By application of TDD in CPPS engineering environments, the domain experts need to specify test cases before the system improvements implementation to support positive risks or to mitigate or avoid negative risks. By application of the Equivalence Class Partitioning (ECP) testing technique, the quality experts reduce the test data space.

The introduction to the thesis has been given in Chapter 1. Chapter 2 presented the related to the problematic work, which revealed research gaps and allowed to elicit the requirements for the designed artefact. Then Chapter 3 described the research approach which allows to design, implement and evaluate the solution which helps the stakeholders to reach their goals. This chapter aims to answer stated in Section 3.1 research questions by introducing *Multi-Aspect Test Case Specification (MATCS) method*. The method is designed in order for quality experts to focus on most relevant tests for cause-effect hypotheses validation to conduct with limited resources, considering the software part of the software intensive systems which are developed/used in multi-disciplinary engineering environments.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Section 4.1 aims to describe detailed requirements which have been initially elicited based on the literature review [Biffl et al., 2017a, Meixner et al., 2020]. However, early workshops with the stakeholders involved in the case studies where the initial set of requirements had been presented, revealed additional requirements for the domain specific applications. The requirements in Section 4.1 have been then extended with the requirements from the domain experts. Based on the detailed requirements, the MATCS method has been designed. Section 4.2 gives an overview of the MATCS method steps, including inputs and outputs for each of the three steps: 1) PPR asset networks; 2) Extended multi-view risk assessment; 3) Multi-aspect test case specification. Section 4.3 introduces the first step of the method process - building of *PPR asset networks*, mostly based on Winkler et al. [2021]. Second step of the method process, *multi-view risk assessment*, is described in section 4.4. This step is an extended version of Biffl et al. [2021]. *Risk drivers identification sub-step*, based on ISO25030 [2019], Antão et al. [2018] and Felderer and Schieferdecker [2014], gives an answer to the first research question: RQ1) which risk drivers motivate multi-aspect risk-based testing of cause-effect hypotheses between production process characteristics and production resource parameters/influences? Following, *risk regions identification* step, identified as mandatory by the domain experts, has been added as an extension to the 4.4. Section 4.5 *multi-aspect test case specification* is designed to build yet another layer of abstraction next to the PAN method and multi-view risk assessment with the help of multiple testing techniques and methods. This part answers the remaining research questions: RQ2) what knowledge model can represent multi-aspect cause-effect hypotheses between production process characteristics and production resource parameters/influences? RQ3) what are the process steps to support multi-aspect testing in CPPS and Software engineering? RQ3a) what is the minimum set of test cases and test data a test engineer requires to validate a hypothesis based on cause-effect relationships? RQ3b) what are the test scope and limitations for multi-aspect test automation? Section 4.6 is designed to present an application of the MATCS method in a generic example.

4.1 Requirements Elicitation

This section is designed in order to elicit the detailed requirements. The design science cycle aims at answering the following question:

How to <(re)design an artifact>
that satisfies <requirements>
so that <stakeholder goals can be achieved>
in <problem context>? [Wieringa, 2014]

In other words: if the designed solution satisfies the proposed requirements, the solution would contribute to the achievement of the stakeholder's goals described in Section 1.1. In context of this thesis, the question is following:

How to design and evaluate a method that facilitates reproducible data collection as foundation for validating the cause-effect hypotheses between process characteristics

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

and resource parameters/influences so that quality assurance specialists from different engineering disciplines can derive and automate multi-aspect test cases in the context of production/transformation processes that depend regarding their feasibility, performance, and risk on production/transformation resources?

In order to answer the above stated question, the following requirements have been defined by the domain experts during preliminary workshops. Some of them were later involved in the two case studies for two use cases that represent the usual process of test case derivation in software and CPPS engineering domains. While the case studies were ongoing, several workshops and discussions held place to match the requirements and present the results. Based on the initial set of requirements presented in Section 3.1, the following detailed requirements have been derived.

4.1.1 Multi-aspect test case specification meta-model and model requirements

This subsection presents the MATCS meta-model and model requirements.

R1.1 Representation of cause-effect relationships and testing specific visual elements and links between them. For representing the assets which are relevant for a use case, such as products, processes and resources, PPR knowledge representation is needed. Furthermore, FMEA elements related to the PPR assets need to be depicted. Following, cause-effect path has to be visible and easy to follow. The last component are testing specific knowledge elements. All the aspects need to be logically interconnected. This requirement assures easily understandable representation of multi-view knowledge, which allows identification of elements by experts and non-experts in a unified model.

R1.2 Representation of testing knowledge. The stakeholders which possess knowledge needed for the test case specification should be able to store their assumptions on FMEA and PPR assets, related to the test parameters, such us: information regarding the equivalence classes for different effects, levels of measurement, control and execution.

R1.3 Knowledge querying and iterative knowledge extension. The domain experts should be able to query existing knowledge models, in order to retrieve already existing knowledge. Furthermore, it should be possible to extend or refine knowledge in iterations. This means that there should be a possibility to extend the initial basic model with further elements representing system assets . This should allow experts and non-experts to achieve the right level of details in their knowledge representation.

4.1.2 Multi-aspect test case specification method requirements

This subsection concerns MATCS method requirements.

R2.1 Key concepts. The method key concepts should be represented in an easily readable structured generic way, so experts and non-experts can follow and apply them.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

R2.2 Risk assessment and prioritization. Different system requirements present different test challenges. In order to solve such challenges, various approaches are available. This requirement is designed in order to help domain experts to map the investigated risks to system requirements. In other words, it should allow the domain experts to identify the requirements which drive the risks. Following, the experts should be able to define the test strategy and identify appropriate PPR assets and their properties, which could be responsible for the risks. It is very important to identify for which parts of the system, the input data and the input states, it is risky not to define the expected behavior. The designed artefacts which support these requirements should guide the experts towards the most important multi-aspect test cases for the use case related to their multi-disciplinary domain.

R2.3 Support of the equivalence classes partitioning concept. It is important to provide the experts and non-experts with an easily understandable concept for the test data definition and minimization of the test suite. Equivalence classes partitioning [Burnstein, 2003] is a well established technique in software engineering. It is also a suitable tool for production system engineering due to similarity of testing related challenges and its ability to support the domain experts with definition of ranges of data which lead to the same test case result.

R2.4 Test case specification. The goal of the method is test case derivation for cause-effect validation and scope definition for the test case automation. In order for stakeholders to understand and apply the method, well known testing techniques and approaches should be integrated into it. One method should define the data values, which should be used for the test cases. This can be achieved with support of the equivalence classes concept (R2.3). Another method concerns test automation - domain experts from different disciplines should be able to understand the method without problems. The method must have graphically represented logical steps and must be easily understandable by experts and non-experts.

R2.5 Test automation support. Very often not all manual test cases can be automated in general. It is especially hard to decide which of the test cases can be automated in multi-disciplinary engineering environments, due to the heterogeneity of implicit expert knowledge and automation tools. In order to support the experts in definition test automation scope and limitations, with regards to multi-aspect testing, an artefact should be defined.

4.2 MATCS Method Overview

Figure 4.1 represents an overview of the novel multi-aspect test case specification method. The goal of the method is to allow multiple engineering stakeholders to specify the efficient set of test cases in multi-disciplinary environments based on various aspects, such as PAN, extended CPPS-RA, risk drivers, in order to validate cause effect-hypotheses, and make their implicit domain knowledge explicit by representing the knowledge in an sufficient way.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

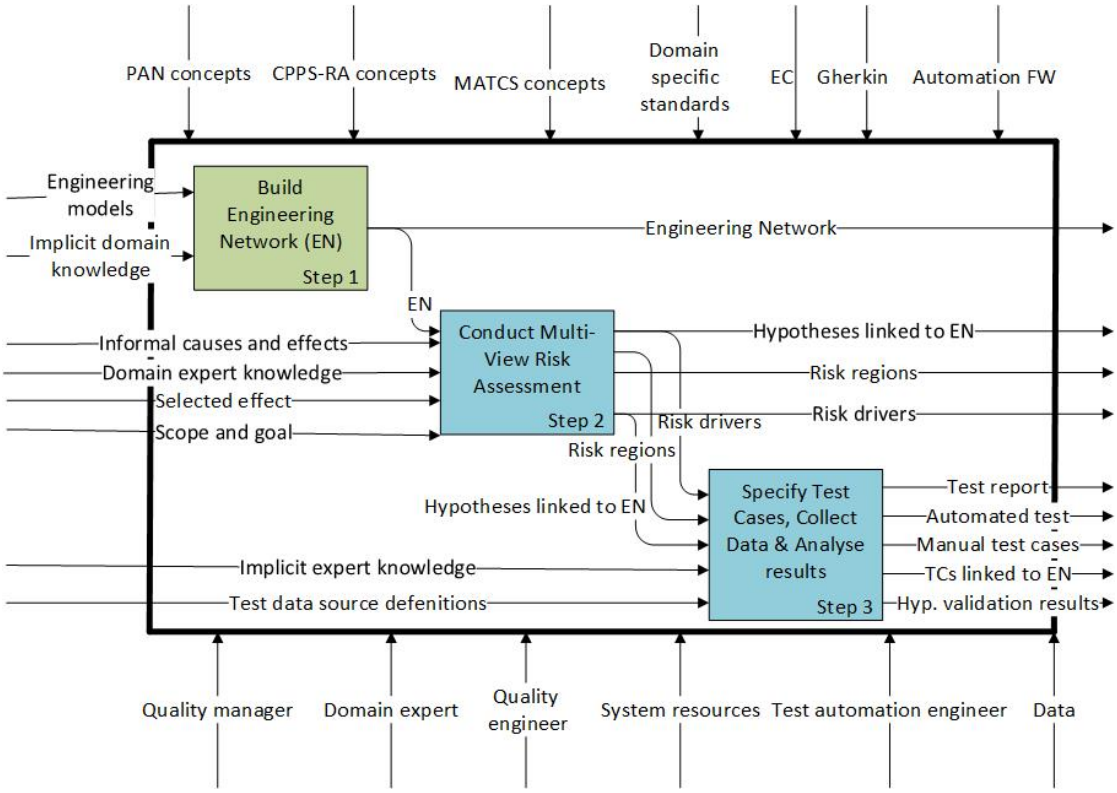

Figure 4.1: Multi-aspect test case specification method overview (in IDEF0 notation [Presley and Liles, 1995]).

In the Figure 4.1, green boxes represent previous work, such as the PAN meta-model and PAN method design, was done by Winkler et al. [2021]. Blue boxes display the main contributions of the thesis and extensions/improvements in the scope of this thesis. The method includes three main steps:

1. Based on various engineering models representing discipline specific views, domain experts build a PAN (Section 4.3). Besides the engineering models, this step requires implicit knowledge from the involved domain experts, as well as PAN meta-model (Figure 2.4) and PAN method (Section 2.1.1).

2. In step two, based on the engineering PAN built in step one, the stakeholders conduct multi-view risk assessment (Section 4.4). This step requires additional knowledge from the experts: informal cause and effect candidates, selected effect to verify, as well as scope and the goal of the risk assessment. The step is supported by the MATCS meta-model (Figure 4.6) and the extended multi-view risk assessment method (Section 4.4). Multi-view risk assessment (also CPPS-RA) method contains two novel sub-steps: 1) identification of risk drivers (Section 4.4, Step 2.1). This sub-step is designed in order to support the decision of the domain experts regarding

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the types of hypotheses validation/testing; 2) risk regions identification (Section 4.4, Step 2.2). The sub-step is designed in order to limit the amount of specified test cases. The output of this step are hypotheses about the cause and effect relationships which need validation.

3. The outputs of step two, together with the output of step one are the inputs for MATCS method (Section 4.5). The method is the main contribution of this master thesis. It consists of five sub-steps. The method allows quality engineers to focus on design and automation of multi-aspect test cases in cooperation with other domain experts, in order to validate the cause-effect hypotheses built in the Step two. The method defines test case data collection and analysis in order to finish the process of hypotheses validation. Specified test cases can be used for test-driven system engineering purposes. Each of the method steps is further described in details in the following sections.

4.3 Step 1. Build Product-Process-Resource (PPR) Asset Network

This section is mostly based on Winkler et al. [2021] and describes the degree this step contributes to MATCS approach. The first step towards the main contribution of this work, e.g. test case derivation for hypotheses testing is, creation of an engineering network (EN), also PAN. An EN is a network which connects engineering assets, such as products, process and resources, within the same and different disciplines using links.

Based on the PAN meta-model displayed in 2.4, domain specific EN is built, as described in Section 2.1.1 of the related work chapter. Figure 4.2 displays the main steps described in Section 2.1.1. The initial work of Winkler et al. [2021] focuses on CPPS Engineering and suggests a diagram with multiple experts with knowledge of different engineering disciplines, such as software, mechanical and electrical engineering. However, in order to achieve a higher level of abstraction and do not limit EN only to CPPS Engineering per se, the domain experts are represented by the single actor in 4.2.

For the first step, e.g. asset identification, multiple domain experts from different engineering disciplines, collect various engineering plans and identify assets related to the future network based on the plans and implicit domain knowledge. The output of the first step is a set of assets to be involved for the EN. During the second step, dependencies identification, the domain experts identify relationships between the assets from the first step. The output of this step is a set of links which represents dependencies between the assets. Next step, PAN creation, allows the domain experts to build the desired PAN, based on the assets and links from the previous steps. The output of step three is an EN which needs to undergo the process of validation and verification in the next step. In the last step, validation and verification, quality experts validate and verify the PAN based on the selected QA techniques and experience, and give a feedback to the domain

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

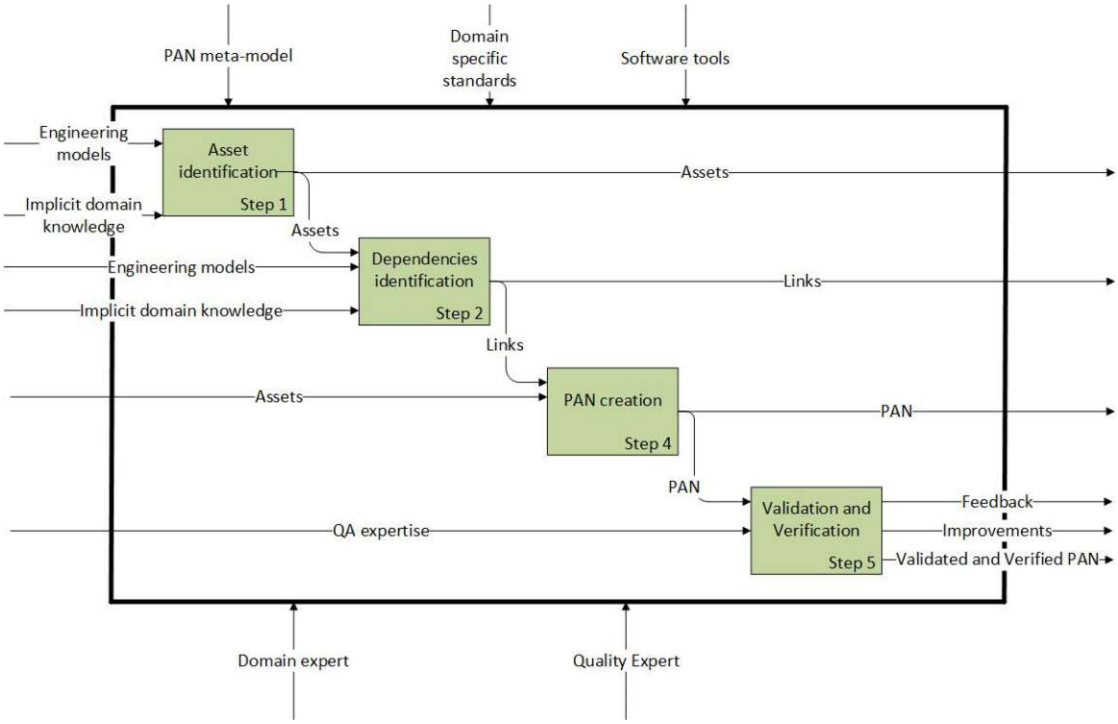

Figure 4.2: PPR asset, dependency elicitation and PAN definition process (in IDEF0 notation [Presley and Liles, 1995]). Based on Winkler et al. [2021].

experts. If, based on the feedback, improvements are required, the whole process or parts of it should be repeated.

4.4 Step 2. Conduct Multi-View Risk Assessment

Once all engineering plans have been consolidated and the EN has been validated and verified, the Multi-View Risk Assessment is about to start.

This method is designed to support domain experts with FMEA based on EN and is an extended version of CPPS-RA from Biffl et al. [2021]. The prerequisite for the method is a domain specific PAN described in Section 4.3 from Step 1. Multi-view risk assessment process is designed to be conducted in four main steps with multiple sub steps with regards to the multi-view risk assessment method described in Section 2.2. The four main steps are displayed in Figure 4.3. Step 2.3 is displayed in a gray rectangle because guidance for this step outside of the scope of this thesis.

Step 2.1. Identify risk drivers This subsection demonstrates the outcome of the analysis of system requirements which require testing, e.g risk drivers, based on ISO25030 [2019], Antão et al. [2018] and Felderer and Schieferdecker [2014].

Quality managers benefit from this outcome, as there is a way to categorize risks and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

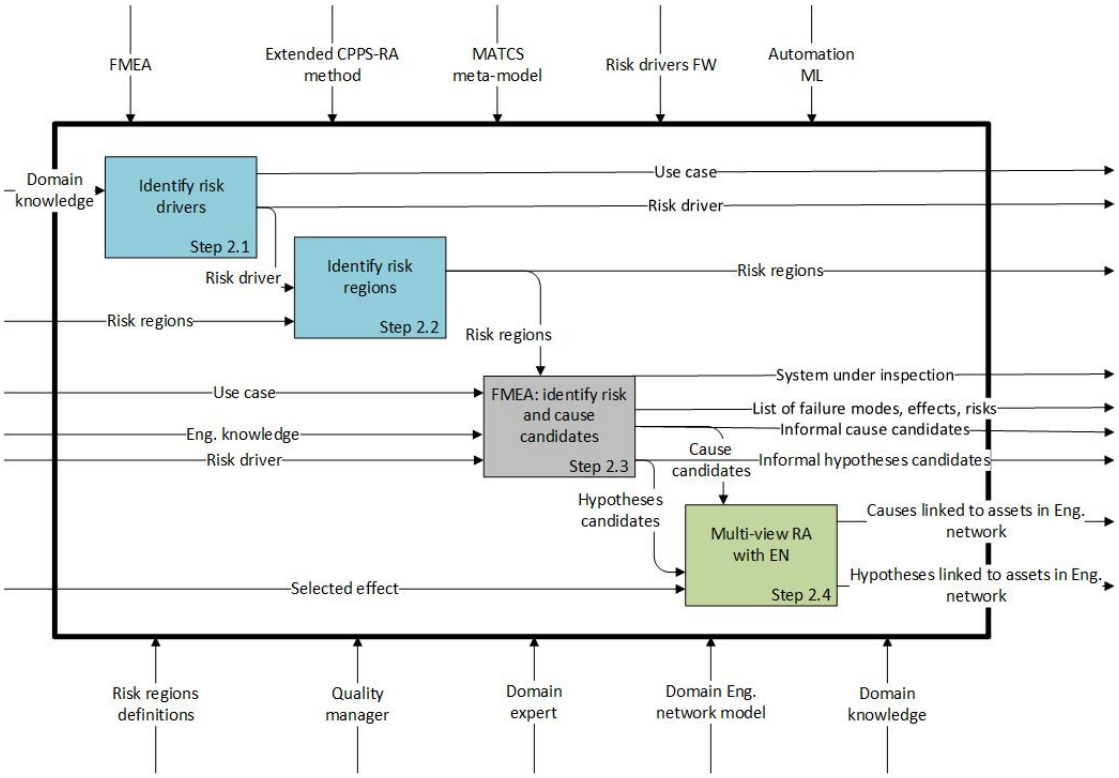

Figure 4.3: MATCS approach overview (in IDEF0 notation [Presley and Liles, 1995]). Based on [Biffl et al., 2020b] and extended.

Risk drivers			
SQuaRE	*CPPS engineering*	*Software engineering*	*MATCS*
Functional suitability		Functionality	Functionality
Performance efficiency	Timing/determinism Scalability		Timing/determinism Scalability
Compatibility	Interoperability		Compatibility
Usability			Usability
Reliability	Reliability, recovery	Reliability Availability Resiliency	Reliability
Security	Security/privacy	Security	Security
Maintainability	Reconfigurability		Maintainability
Portability			Portability
	Safety	Safety	Safety

Figure 4.4: ISO/IEC SQuaRE, CPPS, software systems risk drivers analysis.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

encapsulate the categorization into the multi-aspect testing process. Domain experts and quality engineers benefit from the outcome as well due to the fact that they could use the categories of risks in order to enable better communication and involve the domain experts with the appropriate knowledge. Furthermore, according to the risk drivers, different testing activities can be executed separately. Table 4.4 contains the outcome of the analysis of risk drivers proposed by different standards and authors for multiple types of systems, e.g. CPPS and pure software systems. The analysis has been conducted during workshops with domain experts. The goal of the workshops was to define most critical risk drivers relevant for their domain based on the on ISO25030 [2019] standard and two literature sources: Antão et al. [2018], Felderer and Schieferdecker [2014]. The table is presented in workshops to the domain experts in order to discuss them in regards to real challenges and experiences from daily business. The feedback is incorporated and a set of main risk drivers is concluded in the last column of Table 4.4. The domain experts with CPPS and software engineering background came to a common ground regarding the critical risk drivers. A product or a system is not mature enough to be used if the required functionality is not implemented, therefore functionality is the first common risk driver. Performance of every system is a critical requirement when it comes to the resource utilization, timing. Performance risk driver was described by the domain experts as timing/determinism and scalability. The system and its components need to be compatible with the other systems or components in order to be able to co-exist with each other. Every product or system must be usable by the relevant user. The user of the system and the customers for the end product must trust the system and the quality of the output, therefore every system must be reliable. Proper security is required to protect the assets, services and data from being disrupted or stolen. Not only the development and utilization of a system is important, but also its maintainability, since no system could run forever without changes. In order to transfer a product from one place to another or a module from one system or platform to another, portability is required. In ISO25030 [2019] safety is considered as a non-functional human factor requirement and not mentioned in the list of system requirement which could affect the quality of the product. However, the other two sources mention safety as a system requirement. The domain expert decided to keep safety in the list of system requirements.

Step 2.2. Define risk regions. It is important to know the purpose of the planned risk assessment. The risk assessment approach can be used for prevention of possible undesired risks, avoidance of mitigation of existing risks, assessing and prioritizing possible desired risks which lead to improvements. In order to detect defects or identify possible system improvements, four risk regions and recommendations on how to deal with the values from such regions, have been defined: critical regions to avoid, risky regions to address, regions with a potential for improvements to confirm and regions with a potential for significant improvements to achieve (Table 4.1). Risk drivers definition step provides the first restrictions to the amount of derived test cases.

Step 2.3. FMEA: Identify risk and informal cause candidates.

In the third step, domain experts follow the FMEA process with the goal and scope

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Risk regions	Recommendations
Critical regions to avoid	Introduce FMEA countermeasures to avoid/ detect/deal with the regions
Risky regions to address	Test and introduce FMEA countermeasures to detect/ deal with the regions
Regions with a potential for improvements to achieve	Test to detect and confirm the regions
Regions with a potential for significant improvements to achieve	Build FMEA hypotheses to partition/detect and testing/optimization to confirm/explore the regions

Table 4.1: Risk regions and recommendations.

specified in order to define the system under inspection (SuI). The task of this step is to identify and prioritize candidate failure modes, effects, and risks. For a selected effect E, the domain experts build on the domain specific knowledge to elicit a list of possible cause and hypothesis candidates. The stakeholders use notation such as "Effect E_{1_1} could potentially be caused by Causes C_1, $C_{...}$, C_n or their combinations". This step is visually displayed in Figure 2.8, Step 2.1.

Step 2.4. Software engineering risk assessment with domain specific engineering network

Step 2.4 includes three important sub-steps: 2.4.1) explore EN; 2.4.2) analyze cause candidates and cause-effect pathways; 2.4.3) build hypothesis linked to the EN. This step is visually displayed in Figure 4.5 Step 2.4, and described in related work Section 2.2.2.

Step 2.4.1. Explore Engineering Network.

In this step, the domain experts identify assets that are relevant to represent informal cause candidates. The domain experts start in the EN from the asset that represents the SuI and explore EN assets in an iterative way. The exploration should be conducted by following selected links between assets, which could be related to the selected effect. The expected outcome of this step is a set of links between causes and assets. This step is visually displayed in Figure 4.5, Step 2.4.1 and described in related work Section 2.2.2.

Step 2.4.2. Analyze cause candidates and cause-effect pathways.

The domain experts identify cause candidates linked to the EN elements selected in the risk assessment Step 2.4.1. A cause-effect diagram links the SuI to technically connected assets as foundation for defining a root cause and a pathway that connects the root cause to the SuI and the risky effect. This step is visually displayed in Figure 4.5, Step 2.4.2 and described in related work Section 2.2.2.

Step 2.4.3. Build hypothesis linked to the EN.

The domain experts use a simple restricted language to express their hypotheses based on cause candidates linked to EN elements, e.g., $H(E_{11}; C_1 or C_{...} or C_n)$ (the notation is

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

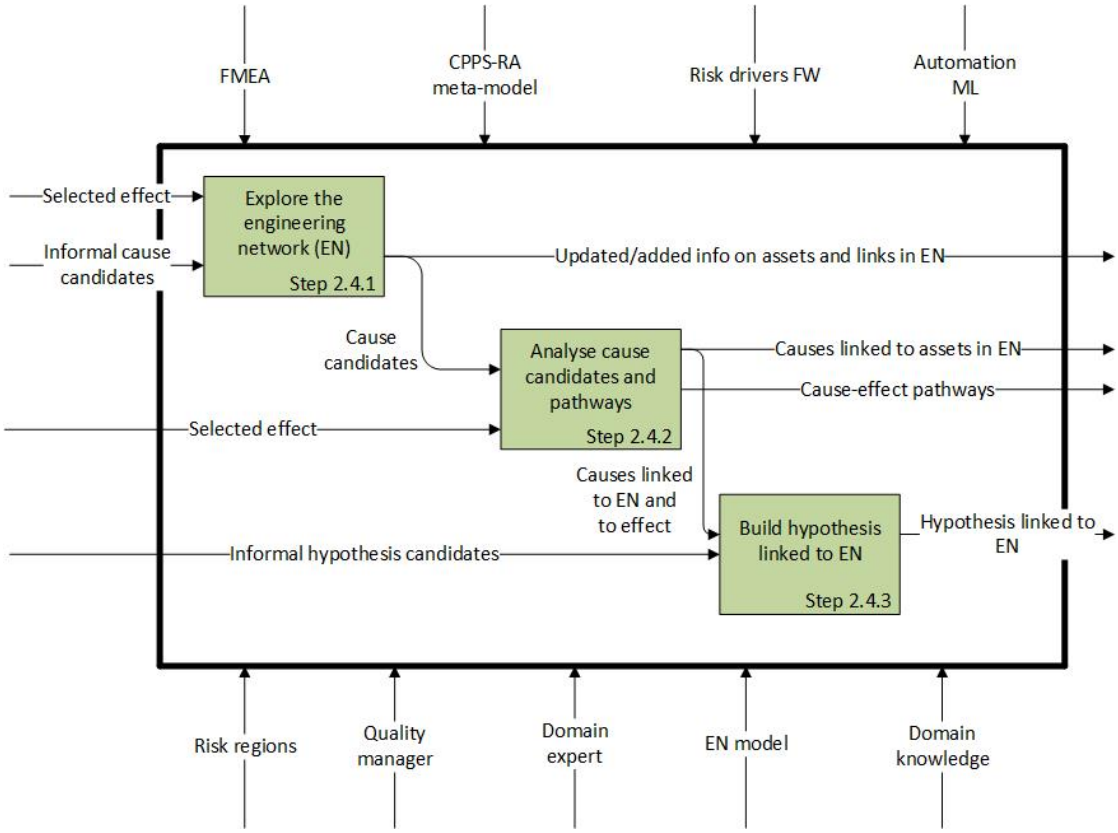

Figure 4.5: Multi-view risk assessment. EN exploration (in IDEF0 notation [Presley and Liles, 1995]). Based on [Biffl et al., 2021].

defined in the Table 4.2). The step is visually displayed in Figure 4.5 Step 2.4.3, and described in related work Section 2.2.2.

The goal of the next section is introduction of a method which would allow the domain experts to derive test cases based on the EN (Section 4.3), the hypotheses, the risk drivers, as well as the risk regions (Section 4.4) identified after conducting the process of multi-view risk assessment.

4.5 Step 3. Multi-Aspect Test Case Specification

This section presents the main contribution of the master thesis - the multi-aspect test case specification method. Figure 4.6 represents the meta-model which is an extension of the CPPS-RA meta-model Biffl et al. [2021]. The meta-model from Biffl et al. [2021] focuses on the connection of core concepts for FMEA, EN, and hypotheses that link effects to causes in the EN. The extended MATCS meta-model depicted in Figure 4.6 contains the test case concept on top of the previous levels, such as FMEA (in violet), EN (in green), and hypotheses that link effects to causes in the EN (in light-blue) and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

test cases (in yellow). The test case concept consists of the following: test case which validates hypotheses, steps which have levels and results which are in fact the effects.

The multi-aspect test case specification method depicted in Figure 4.7 contains five steps: 1) Scoping, 2) Test Case derivation, 3) Test Control and Measurement, 4) Test Automation and 5) Test Data Collection and Analysis. The key concepts of the method are displayed in Table 4.2. The table includes the key concepts for CPPS-RA from Biffl et al. [2021] (lines 1-8) and the concepts related to the new method (lines 9-15). The method uses terminology from [Wohlin et al., 2014] and is designed very similar to Wohlin's Experimentation in Software Engineering. Based on Experimenting in Software Engineering from Wohlin, the following terms for the method are introduced: causes are so-called independent variables, effects are also dependent variables. The cause under investigation is a factor and particular values of factors are treatments. The concepts will be further explained. The following text aims at describing the testing method based on the MATCS testing meta-model in details.

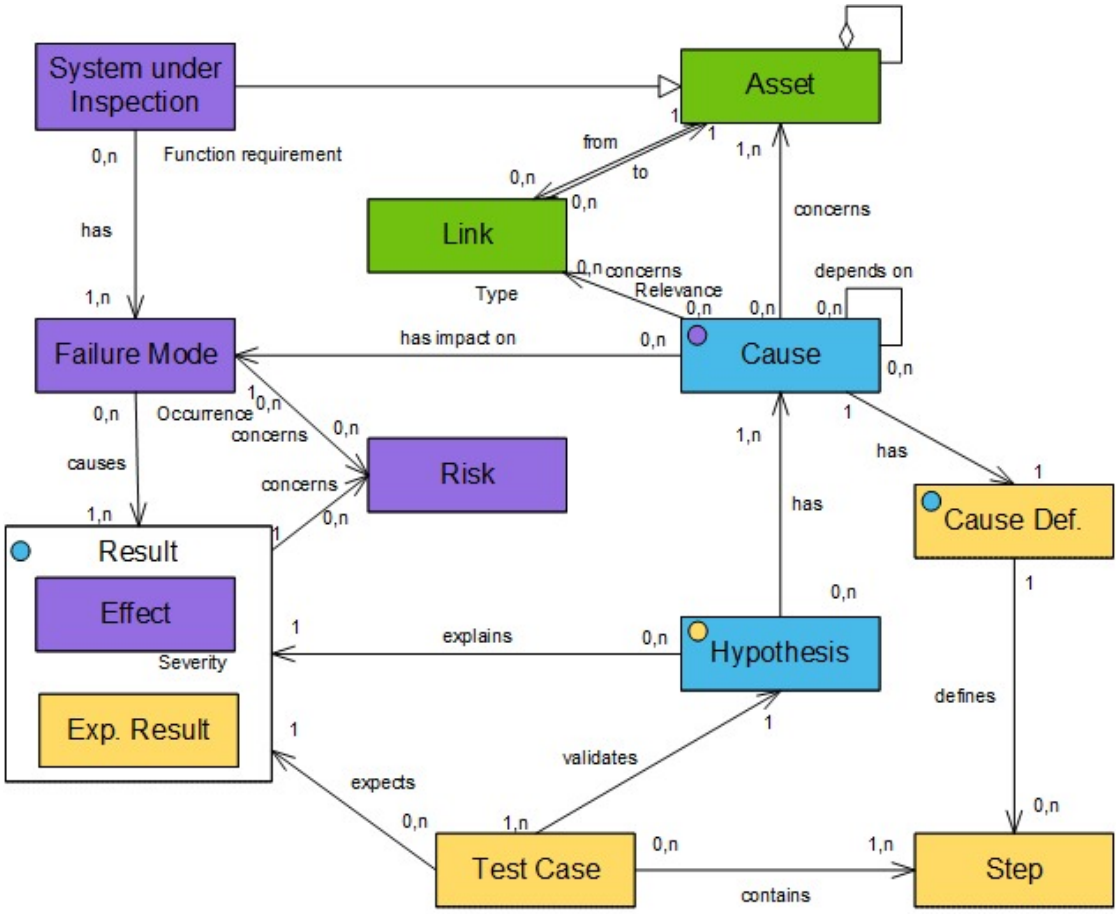

Figure 4.6: Multi-aspect testing meta-model. Based on [Biffl et al., 2021] and extended [Winkler et al., 2022]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

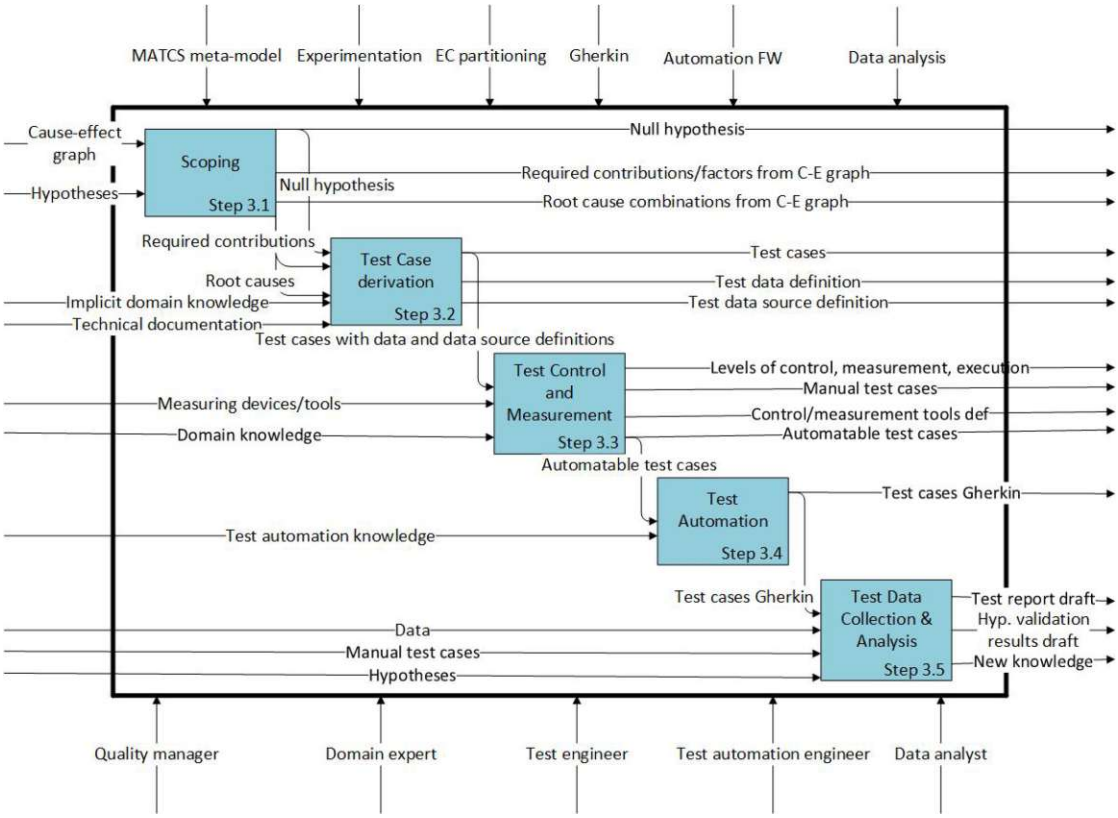

Figure 4.7: Multi-aspect testing MATCS method steps (in IDEF0 notation [Presley and Liles, 1995]).

4.5.1 Step 3.1: Scoping

The first step of the multi-aspect testing method is scoping, which is displayed in Figure 4.7, Step 3.1. The goal of this step is partitioning of the risk regions, building of the Null and at least one None-Null Hypotheses, as well as identification of the number of test cases to be designed.

Building Hypotheses

Null Hypothesis: $H_0(E_{11}; NOR(C_1, C_{...}, C_n))$. This means that none of the following cause candidates is the root cause for the effect.

None-Null Hypothesis: $H_1(E_{11}; OR(C_1, C_{...}, C_n))$, *where* $H_1(E_{11}; C_n! = C_m)$ *for at least one pair of n and m.* This means that at least one of the cause candidates is the root cause of the undesired effect.

The input parameters for this step are: the cause-effect graph from the multi-aspect risk assessment part (Section 4.4) with identified possible causes for specified effects, hypotheses built based on the cause-effect relationships and multi-aspect environment

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Concept	Concept description
SuI	System under Inspection, an Asset
FM_x	Failure Mode x of the SuI
E_{xx}	Effect xx
A;A_F, A_P, A_{P0}, A_R	Asset; a Function (F), Product or Material (P), Process (P'), Resource (R) Asset in a EN
$L_t(A_x, A_y)$	A Link of type t between two Assets, A_x and A_y
$C_x(A_y)$	Cause Cx associated to asset Ay, e.g.,a wrong parameter value leading to a FM.
$H_x(E_{xx}, C_x)$, x!=0	Non Null-Hypothesis, linking effect E_{xx} to a set of causes C_1, ..., C_n via a pathway of assets and links in the EN
$H_x(E_{xx}, C_x)$, x=0	Null-Hypothesis, linking effect E_{xx} to a set of causes C_1, ..., C_n via a pathway of assets and links in the EN
$EC_x(V_x)$	Equivalence Class with a set of single values V_1,..., V_n or values V_1,..., V_n corresponding to intervals
Can measure (A_x)	Measurable value of A_x, possible values: yes, no
Can control (A_x)	Controllable value of A_x, possible values: yes, no
Levels of control (A_x)	Stakeholders who have control over the asset; example values: tester, developer
Levels of execution (A_x)	Env where the test can be executed; example values: test, prod, simulation
Exp. result	Expected output of the test run example values: OK, NOK
Act. result	Actual output of the test run example values: OK, NOK

Table 4.2: Key concepts for multi-aspect test case specification method.

experience of the domain experts. During this phase, the domain experts create a document where they describe the desired and undesired effects, the assets linked to the defects, and the possible causes contributing to the effect. Questions to be answered at the Scoping step:

- Which of the expected effects are desired effects? Which of the causes, with the expected values set, contribute to the effect?
- Which of the expected effects are undesired effects? Which of the causes could lead to the effect?

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- What are the domain specific requirements for the multi-aspect test cases?

The expected outputs of this step are documented: test case specification requirements based on the domain knowledge required contributions/factors from the EN graph and root cause combinations from the EN graph. Following the domain experts iterate through the cause candidate assets from the hypotheses and prepare a draft Table 4.3 for the cause definitions. C_n represents the number of the cause candidate from the knowledge graph, EC_n represents the number of equivalence classes. Each of the asset levels has N equivalence classes. According to the number of causes and equivalence classes, domain experts prepare a table for test cases, Table 4.4.

Critical	Risky	Impr.	Impr++	Measurement	Control	Levels of control	Levels of exec
C_1							
$C_{...}$							
C_n							

Table 4.3: Template Table for Cause Candidate definitions

TC N	C_1	$C_{...}$	C_n	Exp. result	Act. result	H_n	Can control	Can automate
TC 1						H_0		
TC ...						H_1		
TC n						H_1		

Table 4.4: Template Table for Test Cases

4.5.2 Step 3.2: Test Case Derivation

The second step of the method is test case derivation. The input parameters for this step are the results of the scoping step as well as the cause-effect graph, expected results, and equivalence classes (Section 2.3.3) provided by the domain experts. Following, the test engineer, together with the domain experts defines the test data sources. Questions to be answered at the this step:

- What are the equivalence classes for each of the causes and risk regions, and which values are representative for each of them?
- What combinations of values are meaningful for the current scenario?

Once the set of test cases has been derived, the test engineer extends the cause-effect graph with the tests related knowledge such as reference test data and links from the causes to the effects using an iterative approach (Table 4.5, Table 4.6). Since it is often unclear to which equivalence classes which values or ranges of values (equivalence sub-classes) contribute, the process of identification of such regions is often iterative and involves the experimentation method from Wohlin et al. [2014]. The use of the equivalence class

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

partitioning technique, along with the risk regions identification, provides the second round of restrictions to the amount of derived test cases, which makes the final test set efficient.

Critical	Risky	Impr.	Impr++	Measurement	Control	Levels of control	Levels of exec
C_1	V_1	$V_{...}$	V_n				
$C_{...}$	V_1	$V_{...}$	V_n				
C_n	V_1	$V_{...}$	V_n				

Table 4.5: Table for Cause Candidate definitions with Level Values

TC N	C_1	$C_{...}$	C_n	Exp. result	Act. result	H_n	Can control	Can automate
TC 1	V_1	$V_{...}$	V_n	OK		H_0		
TC ...	V_1	$V_{...}$	V_n	NOK		H_1		
TC n	V_1	$V_{...}$	V_n	NOK		H_1		

Table 4.6: Table for Test Cases with Equivalence Classes values

4.5.3 Step 3.3: Test Control and Measurement

The next step of the Method is Test Control and Measurement. At this step, test engineers in cooperation with the domain experts iterates through the set of test cases and adds the info related to the degree of measurement and control to each of the causes. This information is very important since based on it the decision whether it is possible to automate a test case will be made. If all assets can be controlled automatically and execution of such test cases cannot damage the system or the product, the test case can potentially be automated. In case when some of the assets cannot be controlled automatically but can be measured automatically or can only be controlled manually but measured automatically, the test case cannot be automated, however systematic monitoring and data analysis is possible. Questions to be answered at the Test Control and Measurement step:

- Values of properties of which assets can be measured? Who of the stakeholders can measure the values?
- Values of properties of which assets can be controlled? Who of the stakeholders can control the values?
- Where can the test case be executed?

Once the questions have been answered, the tables can be extended: Table 4.7, Table 4.8. In case if an asset value can be controlled on a meaningful level, the action can also be automated.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Critical	Risky	Impr.	Impr++	Measurement	Control	Levels of control	Levels of exec
C_1	V_1	$V_{...}$	V_n	yes	yes	test engineer	Sim
$C_{...}$	V_1	$V_{...}$	V_n	yes	no	test engineer	Prod
C_n	V_1	$V_{...}$	V_n	no	no	external provider	-

Table 4.7: Table for Cause Candidate definitions with Levels of Measurement and Control

TC N	C_1	$C_{...}$	C_n	Exp. result	Act. result	H_n	Can control	Can automate
TC 1	V_1	$V_{...}$	V_n	OK		H_0	yes	yes
TC ...	V_1	$V_{...}$	V_n	NOK		H_1	no	yes
TC n	V_1	$V_{...}$	V_n	NOK		H_1	no	no

Table 4.8: Table for Test Cases with Levels of Control and Automation

4.5.4 Step 3.4: Test Automation

Based on the levels of control of each individual asset, the test engineer makes the decision which of the tests worth automating and whether there is an option for semi-automated test cases in this scenario. At the Test Automation step, the test automation engineer plans test automation activities taking into consideration the levels of control over the causes. The test cases, where each cause can be automatically controlled, could be fully automated. The test cases, where only some of the causes can be automatically controlled, could be semi-automated. Selection of the tools is based on the preferences of individual stakeholders or companies. It is recommended to use Gherkin syntax[1] or a keyword-driven approach [Rwemalika et al., 2019] to describe the tests in a readable for experts and non-experts way and automate them using the preferred automation tool.

4.5.5 Step 3.5: Test Data Collection and Analysis

During the last step, quality expert creates a plan on how to generate test reports. Selection of the tools is based on the preferences of individual stakeholders or companies.

Once the tests output is received, responsible stakeholders can complete the testing table and either reject or fail to reject the stated hypotheses based on the correlation between the expected and actual results. Table 4.9 displays the following example: the experts failed to reject H_0 and H_1, and could not execute one of the test cases due to the unappropriated levels of control. This means that the experts were right about the cause candidates which lead to the effect and no further investigation is needed. If the experts rejected all hypotheses except of H_0, it would mean that the cause candidates were selected incorrectly and another iteration through the whole process would be needed. The test cases which cannot be executed, but their value can be automatically measured, can be included in the future development plans and support the Test-driven system engineering approach.

[1]Gherkin syntax: `https://cucumber.io/docs/gherkin/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

TC N	C_1	$C_{...}$	C_n	Exp. result	Act. result	H_n	Can control	Can automate
TC 1	V_1	$V_{...}$	V_n	OK	OK	H_0	yes	yes
TC ...	V_1	$V_{...}$	V_n	NOK	NOK	H_1	no	yes
TC n	V_1	$V_{...}$	V_n	NOK		H_1	no	no

Table 4.9: Table with expected and actual test results.

4.6 MATCS Method Application

This section displays a generic example where the MATCS method has been applied. The example and its conceptual has been published in the 27th (IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) [Winkler et al., 2022].

4.6.1 Build PAN.

Figure 4.8 displays a generic example of an EN. After the four steps process described above has been conducted, the EN contains: (1) PPR assets represented by PPR asset network nodes; (2) related asset properties represented by PAN links. On the Figure 4.8, processes, e.g. Process 1 and Process 2 are represented by rectangles and are connected with each other and with other assets, such as products (the circle) and resources (ovals), via Product-Process (full one direction arrows) and Process-Resource (dashed arrows) links. Resources are connected with other resources using custom engineering discipline related links, such as, in this example, Link A (dotted line) and link B (dashed line). The EN is now ready for the next step - multi-view risk assessment.

4.6.2 Conduct multi-view risk assessment.

During this step the domain experts conducted multi view risk assessment described in Section 4.4.

Step 2.1. Identify risk drivers. Assuming that the generic example concerns a system scalability undesired effect, scalability test cases need to be specified. The purpose of scalability testing is making sure that the system can handle an increased or decreased user traffic.

Step 2.2. Identify risk regions. The scalability undesired effect is considered as risky. In this scenario, the goal is to identify the asset and its property which could lead to the undesired effect. Therefore only two regions are considered: default region with values when the system is expected to work as designed and the risky region - the region which should be confirmed and FMEA countermeasures need to be introduced.

Step 2.3. FMEA: Identify risk and informal cause candidates. At the third step of the multi-view risk assessment domain experts identify possible risks and their cause candidates. The possible risk is the scalability issue. A list of informal cause and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

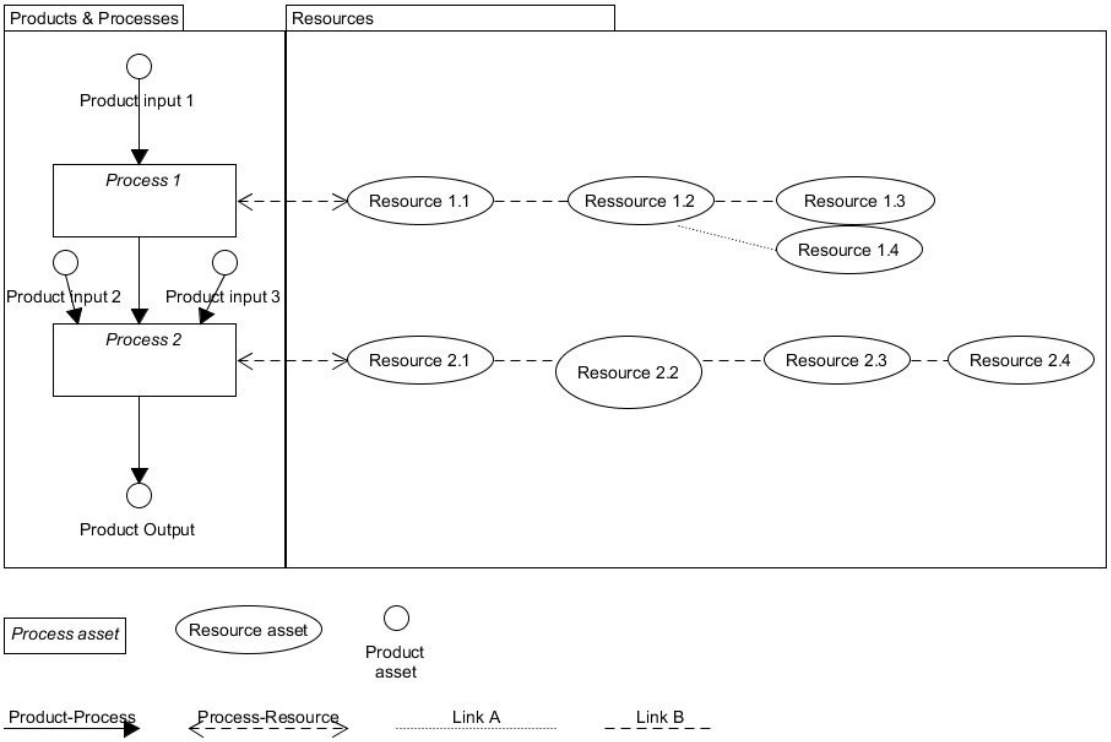

Figure 4.8: Generic product-process-resource asset network example.

hypothesis candidates is delivered after this step. In the current example the undesired effect happens due to a problem in resources related to Process step 1 or Process step 2.

Step 2.4. Software engineering risk assessment with domain specific engineering network. The three steps related to software engineering risk assessment with domain specific engineering network are presented here.

Step 2.4.1. Explore engineering network.

Figure 4.9 represents a generic example for the three sub-steps of the the multi-view risk assessment process. In order to identify cause candidates for the undesired effect which happens to the SuI, the domain experts explore the EN in an iterative way. In the current example, there are three possible pathways of exploration for the domain experts, such as:

1. S_0 (SuI) -> S_1 (Process 1) -> S_2 (Product input 1) -> S_3 (Resource 1.1) -> S_4 (Resource 1.2) -> S_5 (Resource 1.3),

2. S_0 (SuI) -> S_1 (Process 1) -> S_2 (Product input 1) -> S_3 (Resource 1.1) -> S_4 (Resource 1.2) -> S_5 (Resource 1.4),

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

3. S_0 (SuI) -> S_1 (Process 2) -> S_2 (Product input 3) -> S_3 (Resource 2.2) -> S_4 (Resource 2.4)

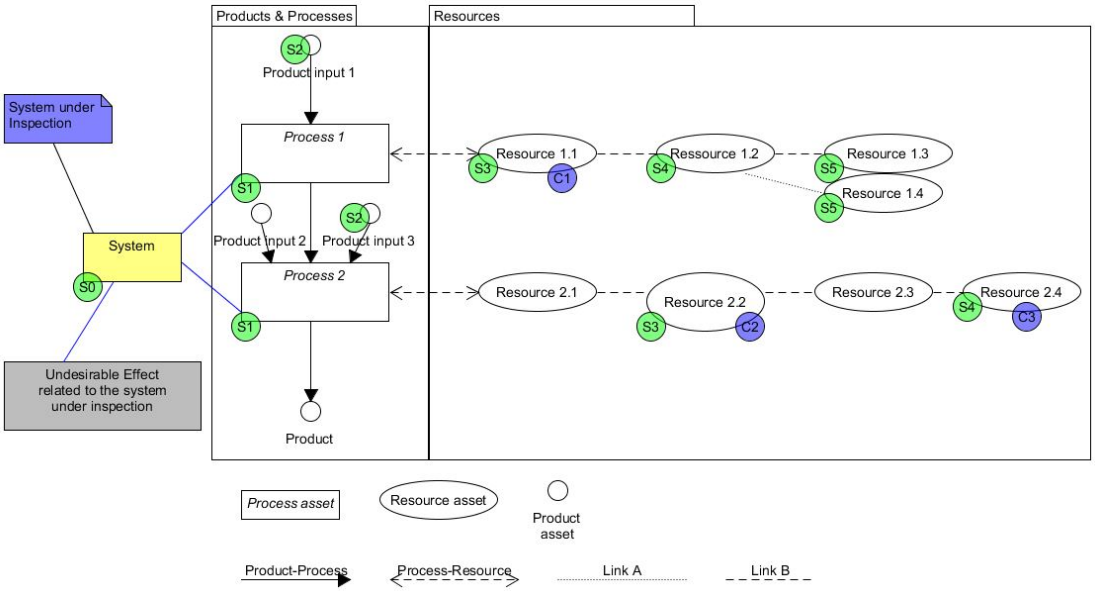

Figure 4.9: Generic product-process-resource asset network exploration with cause candidates example.

However, based on the opinions of the domain experts, not each asset from the path is a cause candidate for the undesired effects. On the pathways 1. and 2. only Resource 1.1 is a cause candidate, while on the pathway 3., there are two cause candidates: Resource 2.2 and Resource 2.4. The cause candidates are market as C_1, C_2 and C_3 on the Figure 4.9.

Step 2.4.2 Analyze cause candidates and cause-effect pathways.

Figure 4.10 represents an FMEA + PPR diagram displaying all three possible pathways from the cause candidates to the undesired effect.

Figure 4.9 displays an example of an artefact created during the multi-view risk assessment process. On top of the PAN, a SuI is added, e.g. system. The system is inspected for the case of possible undesired effect. The effect can be caused by resources coming from different parts of the system and different disciplines or the environment, e.g. cause candidates C_1, C_2, C_3.

Step 2.4.3. Build hypothesis linked to the EN

A hypothesis built based on the knowledge about the cause candidates, describes it in the following way: $H(E_{11}; C_1 or C_2 or C_3)$. This means that the undesired effect, based on the assumptions from the domain experts, could happen because of an issue in Resource 1.1 or in Resource 2.2 or in Resource 2.4.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

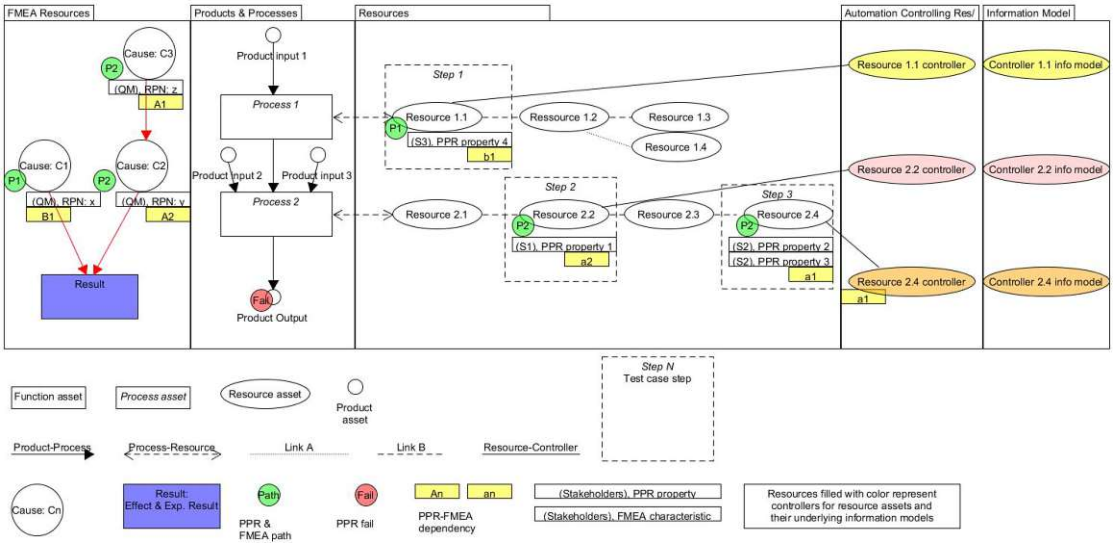

Figure 4.10: FMEA + PPR: cause-effect pathway (red arrows) leading from causes (white circles) to an FMEA effect (violet box) via linked assets in an example engineering network (green circles), including the test case steps (dashed rectangles) [Winkler et al., 2022]

4.6.3 Specify test cases.

The output of after the PAN and CPPS-RA processes is: $H(E_{11}; C_1 or C_2 or C_3)$. This means that the undesired effect, based on the assumptions from the domain experts, could happen because of an issue in Resource 1.1 or in Resource 2.2 or in Resource 2.4.

Step 3.1. Scoping.

Null Hypothesis: $H_0(E_{11}; NOR(C_1, C_2, C_3))$. This means that none of the following cause candidates is the root cause for the effect.

None-Null Hypothesis: $H_1(E_{11}; OR(C_1, C_2, C_3))$, *where* $H_1(E_{11}; C_n != C_m)$ *for at least one pair of n and m.* This means that at least one of the cause candidates is the root cause of the undesired effect.

Step 3.2. Test case derivation.

Assuming that each of the causes has two equivalence classes EC_1 (OK) and EC_2 (Risky), Table 4.12 represents the cause candidate definition. Values for EC_1, such as for example V_{12}, represent correct values, with such values the system should work as expected. Values in EC_2 represent wrong values. If such values are set, the undesired effect might appear. Wrong values are highlighted with pink in the Table 4.12.

Based on the cause definition, test cases are derived, which are represented by Table 4.13. The amount of test cases is equal to 3^2=8, where 3 is the amount of cause candidates

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	OK	Risky	Can measure	Can control	Levels of control	Levels of exec
C_1 (Resource 1.1)	$Value_{11}$	$Value_{12}$				
C_2 (Resource 2.2)	$Value_{21}$	$Value_{22}$				
C_3 (Resource 2.4)	$Value_{31}$	$Value_{32}$				

Table 4.10: Table for cause candidate definitions with levels of measurement and control for the generic example

and 2 is the amount of equivalence classes for each of the cause candidates. Wrong values are highlighted with pink in the Table 4.13.

TC N	C_1	C_2	C_3	Exp. result	Act. result	H_n	Can control	Can automate
TC 1	V_{11}	V_{21}	V_{31}	OK		H0		
TC 2	V_{12}	V_{21}	V_{31}	NOK		H1		
TC 3	V_{11}	V_{22}	V_{31}	NOK		H1		
TC 4	V_{12}	V_{22}	V_{31}	NOK		H1		
TC 5	V_{11}	V_{21}	V_{32}	NOK		H1		
TC 6	V_{12}	V_{21}	V_{32}	NOK		H1		
TC 7	V_{11}	V_{22}	V_{32}	NOK		H1		
TC 8	V_{12}	V_{22}	V_{32}	NOK		H1		

Table 4.11: Test cases example

Step 3.3. Test control, measurement and execution

Assuming that each of the assets can be controlled and measured by available domain experts, and where the test can be executed. Since each of the causes can be measured and controlled separately, and executed on the test environment, all assets in each of the test cases can be controlled and thus, each of the test cases can be automated.

	OK	Risky	Can measure	Can control	Levels of control	Levels of exec
C_1	V_{11}	V_{12}	yes	yes	test engineer	test
C_2	V_{21}	V_{22}	yes	yes	test engineer	test
C_3	V_{31}	V_{32}	yes	yes	mechanical engineer	test

Table 4.12: Table for cause candidate definitions with levels of measurement and control for the generic example

In such a scenario, when all cases could be executed and automated, a test case reduction based on the domain experts' expertise and common sense hould take place. Here it is visible that expected results of test cases TC 2, TC 3 and TC 5 is NOK even if one of the assets is the cause. This means that in case it is proven that the actual results of TC 2, TC 3 and TC 5 is NOK, TC 4, TC 6, TC 7 and TC 8 can be marked as NOK as well, since all of them contain the values of the cause candidates which lead to the undesired

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

TC N	C_1	C_2	C_3	Exp. result	Act. result	H_n	Can control	Can automate
TC 1	V_{11}	V_{21}	V_{31}	OK		H0	yes	yes
TC 2	V_{12}	V_{21}	V_{31}	NOK		H1	yes	yes
TC 3	V_{11}	V_{22}	V_{31}	NOK		H1	yes	yes
TC 4	V_{12}	V_{22}	V_{31}	NOK		H1	yes	yes
TC 5	V_{11}	V_{21}	V_{32}	NOK		H1	yes	yes
TC 6	V_{12}	V_{21}	V_{32}	NOK		H1	yes	yes
TC 7	V_{11}	V_{22}	V_{32}	NOK		H1	yes	yes
TC 8	V_{12}	V_{22}	V_{32}	NOK		H1	yes	yes

Table 4.13: Test cases example

effect. If the actual result for TC 2, TC 3 or TC 5 is OK, a test cases which contain the wrong values in combination with other wrong values should be executed.

Step 3.4. Test automation.

Once it has been established, which of the test cases can be automated and the test cases have been executed manually, a possible Gherkin[2] script has been prepared.

```
Feature: H0 testing
As a data analyst,
I want to test OK
So I can confirm OK scenario

Given I assume C1,C2,C3 are no root cause
And There are 3 independent variables with fixed level
And I do action
And I run the test
And test status is OK
Then I confirm OK scenario

Feature: H1 testing
As a data analyst,
I want to test NOK
So I can confirm NOK scenarios

Given I assume "treatment factor" is the root cause
And There are independent variables with fixed level "V1" and "V2"
When I apply "alternative treatment" to the factor
And I do "setup cause by definition"
And I run the test
```

[2]Gherkin syntax: https://cucumber.io/docs/gherkin/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
And test status is "NOK"
Then I found the root cause

|treatment factor |alternative treatment |V1 |V2 |action |exp result
|C1 |V12 |setup cause by definition |V21 |V31 |OK
|C2 |V22 |setup cause by definition |V11 |V31 |OK
|C3 |V32 |setup cause by definition |V11 |V21 |OK
```

Step 3.5. Test data collection and analysis.

After the manual execution of the test cases, it has been confirmed that each of the cause candidates independently is problematic and causes the undesired effect. This also means that the combinations of causes lead to the undesired effect. The domain experts H_1 and rejected H_0. No further investigation is needed. Furthermore, FMEA countermeasures have been established.

TC N	C_1	C_2	C_3	Exp. result	Act. result	H_n	Can control	Can automate
TC 1	V_{11}	V_{21}	V_{31}	OK	OK	H0	yes	yes
TC 2	V_{12}	V_{21}	V_{31}	NOK	NOK	H1	yes	yes
TC 3	V_{11}	V_{22}	V_{31}	NOK	NOK	H1	yes	yes
TC 5	V_{11}	V_{21}	V_{32}	NOK	NOK	H1	yes	yes

Table 4.14: Test cases reduced example.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Software Engineering Use Case: Algorithm Performance

This chapter introduces the illustrative use case from the large-scale *Company A with in-house software development*. The company employs more than 20.000 employees in 20 countries worldwide and delivers its services to over 20 million registered online customers. One of the in-house software developments is located in Austria and has around 400 colleagues. Software infrastructure includes nearly one thousand micro services which deliver data to various end products. One of the multiple micro services developed by the software department is a Web service delivering data to the front-end application for the end users. For the purposes of the thesis, together with the domain experts from the Company A, Use case *"Algorithm performance"* has been selected.

Section 5.1 is designed to present the detailed use case analysis. In Section 5.2 the process of building PAN is described. Section 5.3 presents the multi-view risk assessment conducted in cooperation with the domain experts. Section 5.4 addresses the multi-aspect test case specification process within the use case. Finally, Section 5.5 presents the evaluation of the method by the domain experts from the company A.

5.1 Algorithm Performance – Use Case Analysis

The testing task is to assure that the software algorithm responsible for the data delivery works as expected and the Web service delivers the data within an expected time window. If the data is not delivered by the service within an expected time window, visible delays could appear on the front-end of the application. The testing task is necessary since the response time is the critical measurement for customers' satisfaction. In order to save testers' time and effort, a test automation project is in place. The test case results coming from the automated test execution, therefore, are not only dependent on the performance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

of the software algorithm, the performance of underlying Web service infrastructure, e.g., a virtual machine where the service is deployed, but also on the performance of the underlying automation system, e.g., a virtual machine where the test case is executed. The use case illustrates how test engineers cope with the uncertainty related to possible causes of the failure in the current scenario. In order to better understand the challenges, in the following text project stakeholders and their dependencies regarding knowledge sharing are introduced.

Project stakeholders

Quality manager is responsible for delivering of the best quality software with regards to the pre-defined deadlines. In order to assure the quality, the quality manager is involved in all stages of the software development cycle and supports other stakeholders in their tasks.

Hardware engineers are responsible for maintenance and scaling of hardware for the host machines where the virtual machines are set up.

Software engineers develop code for various services and application to be tested by test engineers. Among the other tasks of software engineers are confirmation about the found issues and investigation of their nature.

Leanops engineers help in creation of Jenkins[1] scripts needed for the automation project. They maintain and plan resources for Jenkins virtual machines as well as maintain hosts for the virtual machines.

Test engineers design, implement and execute manual test cases as well implement test cases with regards for acceptance criteria from the automation tasks. Among the other tasks of test engineers are analysis of Jenkins nightly report and send an e-mail to all stakeholders

Test automation engineers test automation infrastructure to allow test engineers to implement test scripts. They control status of the assigned automation stories, identify dependencies and blockers, maintain the automation framework. Among the other tasks of test automation engineers are performance of code reviews and follow up on feedback implementation.

Figure 5.1 illustrates the stakeholders involved in the automation project, their interactions and dependencies. Stakeholders are represented by person-like icons of different colors. The arrows corresponding to the color of a stakeholder represent information requested by the stakeholder from the others. The incoming arrows represent the response from the others. In order for a test engineer to find out why the data could not be delivered by the consumer service within an expected time window, knowledge from other stakeholders is necessary. As was already mentioned before, performance of the test case results coming from the automated test case execution can depend not only on a virtual machine where the service is deployed, but also on the performance of the underlying automation system,

[1] Jenkins: `https://www.jenkins.io/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

e.g., a virtual machine where the test case is executed. Therefore, at Company A, the tester needs to ask software engineers, leanOps engineers and hardware engineers about various influences which could cause the issue. Following, the tester reports the results to the quality manager.

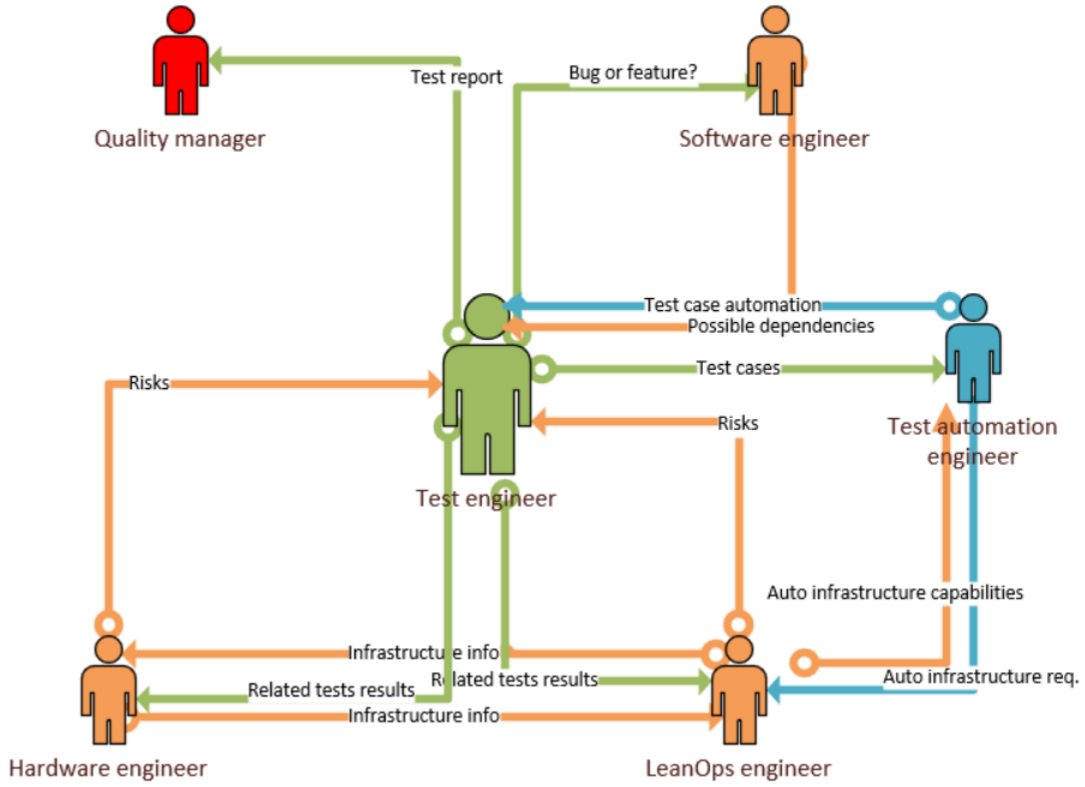

Figure 5.1: Company A - Test automation project stakeholders.

As-Is-Analysis: Test case specification without MATCS method

In order for the domain experts to identify causes of potential risks, information from different domain experts, various reports from the past and technical specifications are to be collected. Once the information has been collected, the domain experts can start searching for the cause of the potential risks. The main challenge here is the fact that the knowledge on underlying systems is distributed among stakeholders which are employed in different teams and different countries. They are not always available when the tester needs their support due to different time zones, vacations etc. The tester contacts different stakeholders according to the past communication experience regarding the possible causes of the problem and gets re-directed to other domain experts or has to wait for experts' availability. Often the times and effort spent on such analysis is unacceptable due to the agile nature of the company Software development life cycle. Meanwhile the issue can get to production and undermine customer confidence and the company's image.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Therefore, in the current scenario, there is often no way for a tester to find the cause which causes the timeout in a meaningful time frame.

Once the possible causes have been identified, testers can design or extend the existing test cases. The existing test cases are located in different test management systems across the company. Testers often do not have access to such artifacts and need to involve domain experts, which leads again to communication and lack of knowledge challenge. If the test cases do not exist yet, they should be created. In order to create such test cases, domain experts should be involved even more, as only they know what levels and values of the causes could trigger the issue. This information is also sometimes documented at places which testers from different departments cannot access.

The collected information on causes, their levels and values is then used for test case design or updates. The information is often limited and enough for a happy path tests. Testers use Ad-hoc testing and equivalence partitioning testing techniques. Such tests allow to verify the state of the system at the particular moment of time, since such test cases are not shared across departments and not maintained. Test data is often created in order to test the functionality limited to one team or department. The influences coming from the outside are mocked.

Execution of such test cases is a big challenge by itself. Testers often do not have the access rights to modify or even view the values of causes from outside their departments or even their teams. In order to execute such test cases, multiple domain experts are involved and long meetings or workshops are organized. The results of the test execution are further analyzed and reported. However, such testing is only done upon demand and not regularly.

Automation for such tests currently does not exist at Company A.

5.2 Algorithm Performance – PPR Asset Network Definition

The main focus of this section is presentation of the PAN implemented by the domain experts from the use case description. The PAN consists of two input and one output products, two process steps and seventeen resource assets. The domain experts have identified four types of links representing relationships between the network assets: Product-Process links, Process-Resource links, Hardware and Software links. The steps for building a PAN are described in Section 2.1.1 and depicted in Figure 4.2. The role of the PAN in test case specification is displayed in a more generic Figure 4.1. Figure 5.3 displays the software engineering PAN representing the use case relevant set of products, processes and resources, built by the domain experts based mostly on their professional knowledge and engineering plans. The domain experts have previous experience with UMLet[2] tool, therefore the tool has been used for the graphic design

[2]UMLet tool: `https://www.umlet.com/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

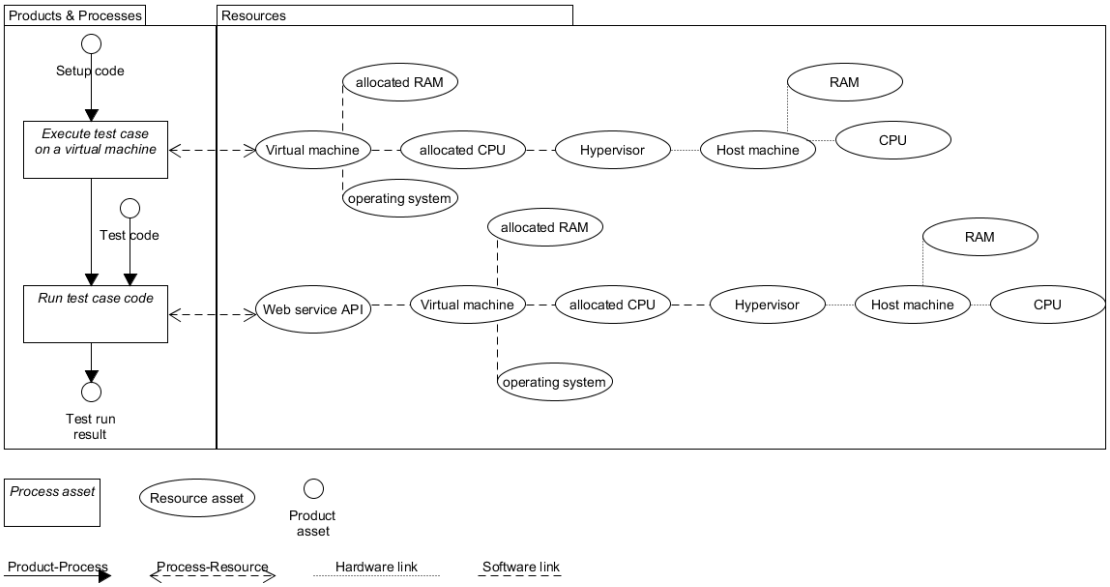

Figure 5.2: Company A - Software engineering PPR Asset Network (PAN).

of the PAN. Following, the knowledge model has been implemented in Neo4j software[3]. The code of the final knowledge model can be found in Appendix A 8.2.

5.3 Algorithm Performance – Software Multi-View Risk Assessment.

This section illustrates how domain experts coming from different departments and disciplines built a *Software Engineering Network (SEN)* based on the CPPS-RA meta-model [Biffl et al., 2021]. Following the experts identified related aspects in the SEN (Figure 5.3) which potentially could contribute to the response timeout. In order to do so they analyzed product, process, resources (as building blocks of the model), and extended the basic model with causes, effects, and dependencies. The extended SEN consists of effects to be observed, related production processes and system resources, influenced by a set of root causes (see details in the legend of Figure 5.3).

The multi-aspect risk assessment Process was conducted in five main steps with regards to the multi-view risk assessment method [Biffl et al., 2021] and the extra steps described in 4.4.

Step 2.1. Identify risk drivers. Within this step of the multi-view risk assessment process, Table 4.4 has been presented to the domain experts. Based on the nature of the specified risk and own expertise, the domain experts came to a conclusion that the presented system requirement is performance, and therefore the risk driver is *Performance*

[3]Neo4j tool: `https://neo4j.com/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

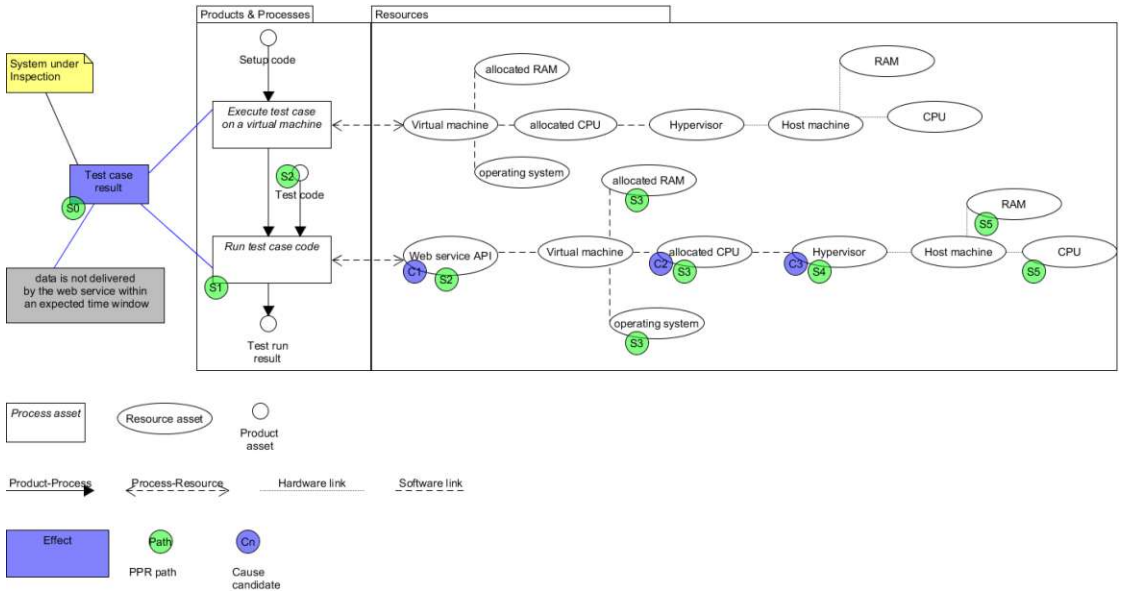

Figure 5.3: Company A - Multi-aspect cause-effect graph.

risk driver. Based on Felderer and Schieferdecker [2014] performance is not a risk driver by itself, however, it is a part of the requirement for reliability of the system. Therefore performance test cases need to be specified.

Step 2.2. Identify risk regions. The insufficient performance of the software algorithm is a risk. In this scenario, the goal is to identify the assets and their properties which lead to the undesired effect. Therefore two risk regions are considered: default region with values when the system is expected to work as designed and the risky region - the region which should be confirmed and FMEA countermeasures need to be introduced. The regions are represented by equivalence classes during the test case specification process.

Step 2.3. FMEA: Identify risk and informal cause candidates. In this step, domain experts followed the FMEA process with the goal and scope specified in order to define the SuI. The task of this step was to identify and prioritize candidate failure modes, effects, and risks. For a selected effect E, the domain experts built on Software Engineering knowledge in the project to elicit a list of possible cause and hypothesis candidates. The stakeholders used notation such as "Effect E_{11} could potentially be caused by causes C_1, C_2, C_3 or their combinations." This means that for the risk *Web service response timeout*, the domain experts identified as possible cause candidates an issue in the Web Service API or insufficient allocated CPU on the Virtual Machine (VM) where the test usually runs or insufficient CPU allocated on the hypervisor for the VM where the test usually runs.

Step 2.4. Software engineering risk assessment with Software engineering network. Step 2.4 includes 3 important sub-steps: 1) explore software engineering network; 2) analyze cause candidates and cause-effect pathways; 3) build hypothesis

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

linked to the SEN.

Step 2.4.1. Explore Software engineering network. In this step, the domain experts identified assets which are relevant to represent informal cause candidates. The domain experts started in the SEN from the asset which represents the SuI. The following query helped the to fetch all assets linked to the SuI:

```
MATCH (n:Product:PPRAsset {name:"Test run result"})-[*]-(m)
RETURN n,m
```

Following, they explored SEN assets in an iterative way. The exploration happened by following selected links between assets, e.g., hardware or software links, which could be related to the selected effect. The domain experts identified eight pathways:

1. S_0 (Test case result) -> S_1 (Test code),
2. S_0 (Test case result) -> S_1 (Web service API),
3. S_0 (Test case result) -> S_1 (Web service API) -> S_2 (allocated RAM),
4. S_0 (Test case result) -> S_1 (Web service API) -> S_2 (operating system),
5. S_0 (Test case result) -> S_1 (Web service API) -> S_2 (allocated CPU),
6. S_0 (Test case result) -> S_1 (Web service API) -> S_2 (allocated CPU) -> S_3 (Hypervisor),
7. S_0 (Test case result) -> S_1 (Web service API) -> S_2 (allocated CPU) -> S_3 (Hypervisor) -> S_4 (RAM),
8. S_0 (Test case result) -> S_1 (Web service API) -> S_2 (allocated CPU) -> S_3 (Hypervisor) -> S_4 (CPU).

However, not all explored assets are cause candidates. The domain experts selected Web Server API, allocated CPU and Hypervisor as cause candidates. The result of this step is a set of links between causes and assets. At this point, the domain experts found the result set of SEN elements sufficient to motivate relevant cause candidates.

Step 2.4.2. Analyze cause candidates and cause-effect pathways. The domain experts identified cause candidates linked to the SEN elements found in the risk assessment Step 2.1, such as an issue in the Web Service API or insufficient allocated CPU on the Virtual Machine (VM) where the test usually runs or insufficient CPU allocated on the hypervisor for the VM where the test usually runs. A cause-effect diagram (Figure 5.4) links the SuI, a Web service response timeout, to technically connected assets as foundation for defining a root cause and a pathway that connects the root cause to the SuI and the risky effect.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

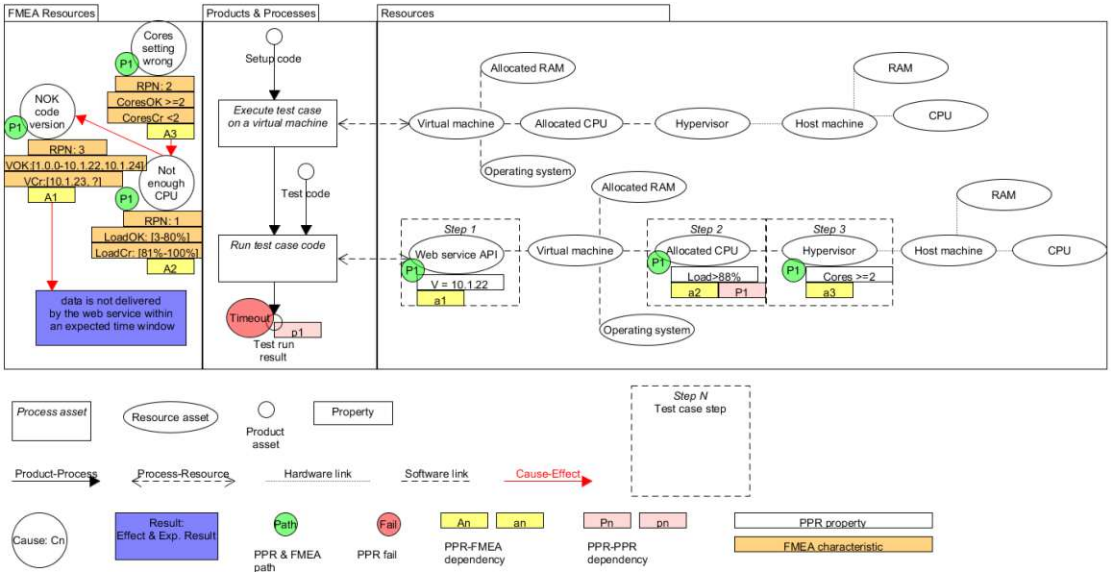

Figure 5.4: Company A - FMEA + PAN: cause-effect pathway (red arrows) leading from causes (white circles) to an FMEA effect (violet box) via linked assets in a SEN (green circles), including the test case steps (dashed rectangles).

Step 2.4.3. Build hypothesis linked to the SEN. The domain experts used a simple restricted language to express their hypotheses based on cause candidates linked to SEN elements, e.g., $H(E_{11}; C_1 or C_2 or C_3)$, where E_{11} represents the effect Web service response timeout, C_1 is issue in the Web Service API, and C_2 is insufficient allocated CPU on the VM where the test usually runs, and C_3 is insufficient CPU allocated on the hypervisor for the VM where the test usually runs.

5.4 Algorithm Performance – MATCS Application

This section illustrates how the test engineer from the Software department of the Company A, together with the domain experts applied the multi-aspect test case specification method in order to generate test cases based on the multi-aspect cause-effect knowledge graph prepared by the domain experts. Following, the test activities were planned, executed and evaluated. The process includes five steps according to the definition of MATCS method, which include: 1) scoping; 2) test case derivation; 3) test control and measurement; 4) test automation; 5) test data collection and analysis.

5.4.1 Step 3.1. Scoping

As described by the MATCS method, the input parameters for this step are: the cause-effect graph with identified possible causes for specified effects, hypotheses built based on the cause-effect relationships and multi-aspect environment experience of the domain

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

experts. At this step, the domain experts created a document where they described the desired and undesired effects, the assets linked to the defects, and the possible causes contributing to the effect.

The desired effect, e.g. data is delivered by the consumer service within an expected time window, happens when the system works as expected, e.g. all the assets, contributing to the effect, correspond to the product requirements.

Cause	EC_1 OK	EC_2 Risky	Can measure	Can control	Levels of control
$C_1 : WebServiceAPI$					
$C_2 : AllocatedCPU(VM)$					
$C_3 : HypervisorSetting$					

Table 5.1: Company A - Template table for cause candidate EC values

TC N	C_1	C_2	C_3	Exp. result	Act. result	H_n	Can control	Can automate
TC 1								
TC 2								
TC 3								
TC 4								
TC 5								
TC 6								
TC 7								
TC 8								

Table 5.2: Company A - Template table for test cases

The undesired effect, e.g. data is not delivered by the consumer service within an expected time window, happens either when one or more of the identified cause candidates is the root cause or none of the identified cause candidates is the root cause, and the root cause is at a different asset or a group of assets. The tester, together with the domain experts, built two hypotheses based on the above stated information:

Null Hypothesis: $H_0(E_{11}; NOR(C_1, C_2, C_3))$. This means that none of the following cause candidates is the root cause for the effect: an issue in the Web Service API or insufficient allocated CPU on the VM where the test usually runs or insufficient CPU allocated on the hypervisor for the VM where the test usually runs. Another representation: P(duration) != f (code issue, CPU load, hypervisor setting).

None-Null hypothesis: $H_1(E_{11}; OR(C_1, C_2, C_3))$, *where* $H_1(E_{11}; C_n! = C_m)$ *for at least one pair of n and m.* This means that at least one of the cause candidates is the root cause of the undesired effect. Another representation: P(duration) = f (code issue, CPU load, hypervisor setting).

The domain experts went iteratively through the cause candidate assets from the hypotheses and prepared a draft Table 5.1 for the cause candidate level values. C_n represents the number of the cause candidate from the knowledge graph, EC_n represents the number of the equivalence class. The experts defined two possible cause candidate levels per asset,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

therefore the Test case template Table 5.1 contains $2^3 = 8$ rows. This means that each of the three asset levels has two equivalence classes: the cause variable could either have the specification (EC_1 OK) value or risky value (EC_2 Risky). The template table is illustrated by Table 5.2.

5.4.2 Step 3.2. Test case derivation

In this step the domain experts and the tester filled in the values for cause candidates and the equivalence classes according to their experience, login and technical requirements. The code version of C_1, e.g. the Web service API could be OK or contain heavy computational operations which lead to the delay on the output. The C_3, e.g. hypervisor CPU setting, could be OK or not OK according to the technical requirements. The C_2, e.g. VM CPU, could be OK or overloaded. However, the domain experts did not know with what load the CPU is considered as overloaded. In order to define the percentage of the CPU usage, they needed to conduct an iterative experiment. Each time they simulated different CPU load on the corresponding virtual machine and re-run the test. Figure 5.5 illustrates the output of the experiment. The experiment contained four runs for three CPU load ranges such as 3-6%, 78-81% and 100%. The result data can be viewed in Appendix A 8.2.The target response time of the service is 200 ms. A slight degradation appeared within the second range (>100 ms) and unacceptable degradation withing the third range (>200 ms during 3 out of 4 runs). According to the results, the domain experts decided that 80% CPU load is the critical value for the host server. Mitigation should be applied already at 60% because running service consumes 20% on an average day, which could bring the value to the risk zone. While executing the test case with CPU 3-6% the experts did not observe any significant performance improvement. Lowering the CPU load even more is not feasible, and therefore the improvement did not take place.

Cause	EC_1 OK	EC_2 OK	EC_3 Risky	Can measure	Can control	Levels of control
$C_2 : CPU(VM)$	3-6%	78-81%	100%	yes	yes	domain expert

Table 5.3: Company A - Table for cause candidate with equivalence classes values after experimentation

TC N	C_1	C_2	C_3	Exp. result	Act. result	H_n	Can control	Can automate
TC 1	OK	3-6%	>= 2	[1-200ms]	94ms	H_0		
TC 2	OK	1-81%	>= 2	[1-200ms]	106ms	H_1		
TC 3	OK	100%	>= 2	[200ms+]	353ms	H_1		

Table 5.4: Company A - Table for experimentation with values from equivalence classes

Based on the analysis, the experts could fill in the tables 5.5 and 5.6 with the corresponding values. The first test case is a positive test case, which means that if all the values are set correctly, the timeout does not happen. The other seven test cases are negative test cases, meaning that if at least one of the values is not set correctly, the timeout happens.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

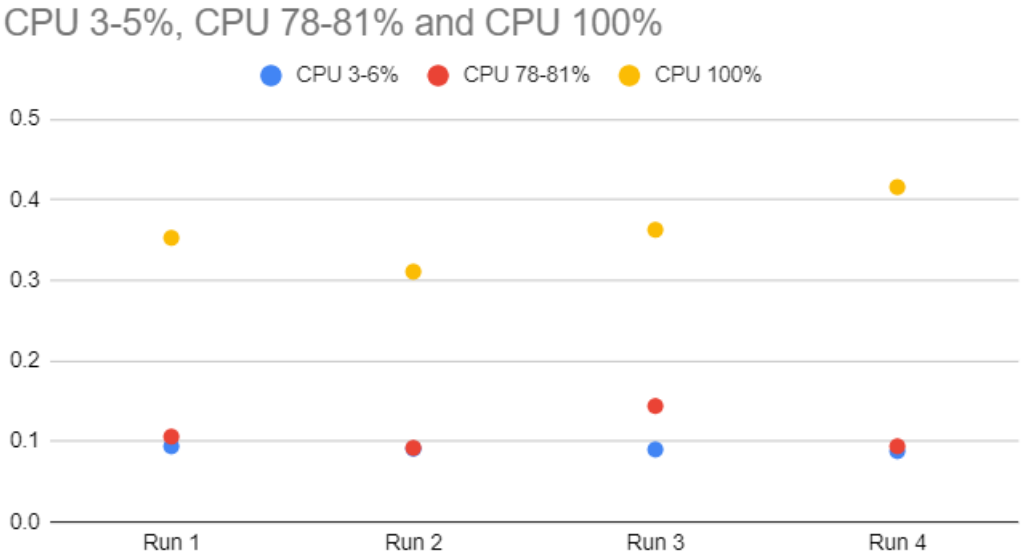

Figure 5.5: Company A - CPU load on the virtual machine during the four runs experiment.

Following, the test engineer, together with the domain experts defined the test data sources, preconditions and post-conditions. Once the set of test cases has been derived, the test engineer extended the cause-effect graph with the tests related knowledge such as reference test data and links from the causes to the effects using an iterative approach.

Cause	EC_1 OK	EC_2 Risky	Can measure	Can control	Levels of control
$C_1 : WebServiceAPI$	V OK	V NOK			
$C_2 : AllocatedCPU(VM)$	<= 80%	> 80%			
$C_3 : HypervisorSetting$	>= 2 cores	< 2 cores			

Table 5.5: Company A - Table for cause candidates with values from identified equivalence classes

TC N	C_1	C_2	C_3	Exp. res.	Act. res.	H_n	Can control	Can automate
TC 1	V OK	<= 80%	>= 2 cores	OK		H_0		
TC 2	V OK	> 80%	>= 2 cores	Timeout		H_1		
TC 3	V NOK	> 80%	>= 2 cores	Timeout		H_1		
TC 4	V NOK	<= 80%	>= 2 cores	Timeout		H_1		
TC 5	V OK	> 80%	< 2 cores	Timeout		H_1		
TC 6	V NOK	> 80%	< 2 cores	Timeout		H_1		
TC 7	V OK	<= 80%	< 2 cores	Timeout		H_1		
TC 8	V NOK	<= 80%	< 2 cores	Timeout		H_1		

Table 5.6: Company A - Table for test cases with equivalence classes values.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.4.3 Step 3.3. Test control and measurement

At this step, the test engineer in cooperation with the domain experts iterated through the set of test cases and defined the level of measurement and control. The following levels of control have been identified:

- tester control: none of the assets. This means the test engineer cannot control any of the presented assets.
- domain experts (involved in the case study) control: the asset corresponding to C_2. The leanOps engineer can control the allocated CPU on the virtual machine by removing / adding CPU cores manually.
- domain experts (all) control: all assets. All properties of all the assets related to this case study can be controlled by one or another domain expert. However, not all of them participated in the study. Therefore the first two points apply.

Cause	EC_1 OK	EC_2Risky	Can measure	Can control	Levels of control
$C_1 : WebServiceAPI$	V OK	V NOK	yes, auto	no	domain expert
$C_2 : AllocatedCPU(VM)$	<= 80%	> 80%	yes, auto	yes	domain expert
$C_3 : HypervisorSetting$	>= 2 cores	< 2 cores	no	no	domain expert

Table 5.7: Company A - Properties of cause candidates.

TC N	C_1	C_2	C_3	Exp. result	Act. result	H_n	Can control	Can automate
TC 1	V OK	<= 80%	>= 2 cores	OK		H0	yes	yes(default)
TC 2	V OK	> 80%	>= 2 cores	timeout		H1	yes	no
TC 3	V NOK	> 80%	>= 2 cores	timeout		H1	yes	no
TC 4	V NOK	<= 80%	>= 2 cores	timeout		H1	no	no
TC 5	V OK	> 80%	< 2 cores	timeout		H1	no	no
TC 6	V NOK	> 80%	< 2 cores	timeout		H1	no	no
TC 7	V OK	<= 80%	< 2 cores	timeout		H1	no	no
TC 8	V NOK	<= 80%	< 2 cores	timeout		H1	no	no

Table 5.8: Company A - Test cases.

The following relevant to the cause asset properties levels of measurement have been identified by the domain experts:

- tester: the assets corresponding to C_1 and C_2. Test engineer can check the version of the deployed service API code and measure the CPU load of the virtual machine automatically.
- domain experts (involved in the case study) measurement: the assets corresponding to C_1 and C_2. LeanOps engineer can check the version of the deployed service API code and measure the CPU load of the virtual machine automatically.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- domain experts (all) measurement: all assets. All properties of all the assets related to this case study can be measured by one or another domain expert. However, not all of them participated in the study. Therefore the first two points apply.

Following, the tester added the info related to the degree of measurement and control to each of the causes (Table 5.7). Based on the results of the measurement and control analysis, the test engineer defined which of the test cases could be executed manually (TC 1 and TC2 in Table 5.8). C_3 was excluded from the consideration due to the fact that it could not be controlled by the domain experts involved in the case study.

5.4.4 Step 3.4. Test automation

Based on the levels of control of each individual asset, the test automation engineer made the decision to automate TC 1. Since the use case concerns a test automation project, TC 1 corresponds to the happy path of already automated test case. Based on the decision of the test automation engineer, keyword-driven test automation approach [Rwemalika et al., 2019] in Robot framework[4] used for the test automation project. There is no possibility for semi-automated test cases in this scenario. The other test cases cannot be automated due to the current lack of control rights over the properties of the network assets. However, control over the CPU load and the hypervisor settings could be implemented in the future in case the domain experts would like to automate the TC 2.

```
*** Settings ***
Resource          ../../../../config/config.txt
Force Tags        cdsapi    Betstation
Suite Setup       Run Keywords
...               setup CDS    AND    CDS setup set TV2 fixture suite variables    AND
...               CDS set fixture dates    start_dateUTC=04:10:00    cutoff_dateUTC=04:55:00    AND
...               CDS setup TV2 greyhounds fixture    fixture_name=greyhoundtoday    start_dateUTC=${start_dateUTC}    cutOff_dateUTC=${cutoff_dateUTC}
Suite Teardown    Tradingv2 greyhounds cleanup fixture    fixtureId=${fixtureIds}

*** Test Cases ***
01 Verify if today results for greyhounds racing events are displayed
    [Documentation]    SPO-17798 - Verify if today results for greyhounds racing events are displayed
                ...    Getting today-result reponse from CDS
                ...    Verify today-result startDate equal to todayDate and isResulted equals to True
    ${response}    Cdsapi greyhounds racing today-result post response verification
    _verify today-result fixture details    ${response}

*** Keywords ***
_verify today-result fixture details
    [Documentation]    Verify that the startDate of the created fixture should be today's date
                ...    Verify that the created fixture IsResulted as True
    [Arguments]    ${response}
    ${date}    Get Current Date    UTC
    ${date}  Convert Date    ${date}    result_format=%Y-%m-%d
    FOR    ${item}    IN    @{response['payload']}
        ${Start_date}    Set variable    ${item['fixture']['startDate']}
        ${Start_date}    Get Substring    ${Start_date}    0    10
        Should Be Equal As Strings    ${Start_date}    ${date}
        Should Be Equal As Strings    ${item['fixture']['addons']['isResulted']}    True
    END
```

Figure 5.6: Company A - a snapshot of test cases written following the keyword-driven test automation approach [Rwemalika et al., 2019] in Robot framework.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TC N	C_1	C_2	C_3	Exp. result	Act. result	H_n	Can control	Can automate
TC 1	V OK	<= 80%	>= 2 cores	OK	OK	H0	yes	yes(default)
TC 2	V OK	> 80%	>= 2 cores	timeout	timeout	H1	yes	no
TC 3	V NOK	> 80%	>= 2 cores	timeout		H1	yes	no
TC 4	V NOK	<= 80%	>= 2 cores	timeout		H1	no	no
TC 5	V OK	> 80%	< 2 cores	timeout		H1	no	no
TC 6	V NOK	> 80%	< 2 cores	timeout		H1	no	no
TC 7	V OK	<= 80%	< 2 cores	timeout		H1	no	no
TC 8	V NOK	<= 80%	< 2 cores	timeout		H1	no	no

Table 5.9: Company A - Test case results

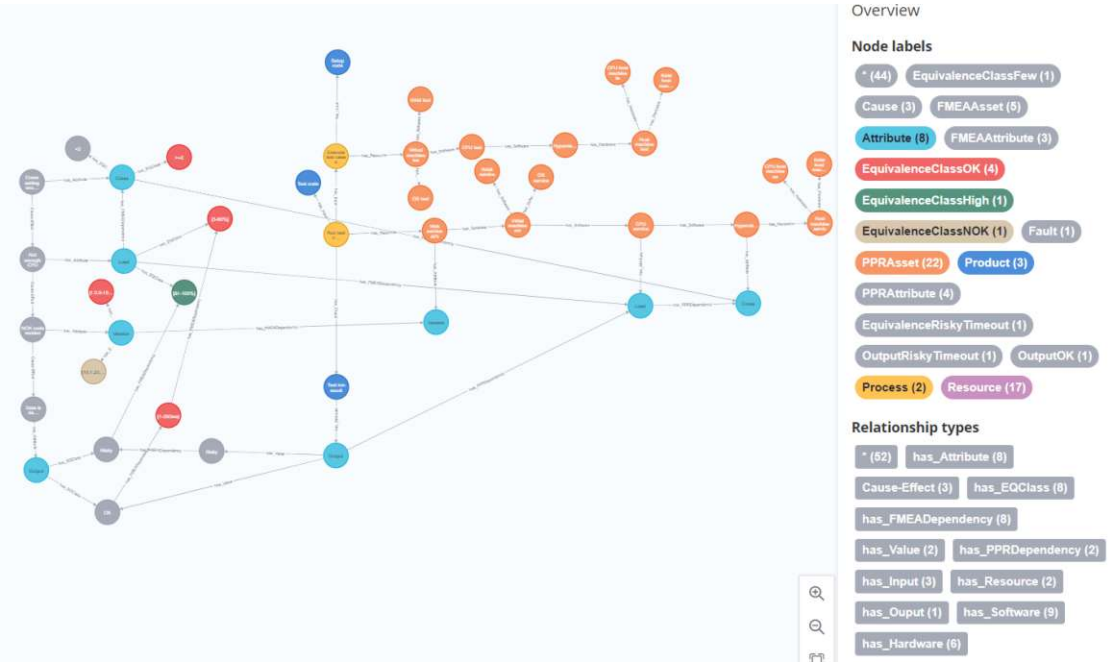

Figure 5.7: Company A: FMEA + PPR + test related knowledge in Neo4j: cause-effect pathway (link cause-effect on the left) leading from causes (gray circles on the left) to an FMEA effect (gray bottom circle on the left) via linked assets in a SEN (orange circles). Equivalence classes are linked to the FMEA attributes of the FMEA assets (circles linked to the blue circles).

5.4.5 Step 3.5. Test data collection and analysis

In the final step, the tester executed the automated and manual test cases and validated the expected results against actual results. Table 5.9 represents the results of the test run. TC 1 was executed automatically and the expected result is equal to the actual result. TC 2 was executed manually with help of the domain experts. In this case, the expected result is equal to the actual result as well. TC 2 due to the current lack of

[4]Robot framework: `https://robotframework.org/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

control rights over the properties of the network assets. However, control over the CPU load and the hypervisor settings could be implemented in the future in case the domain expert would like to automate the TC 2.

After the test runs, the tester could reject H_0 and failed to reject H_1. The root cause of the timeout is C_2, high CPU load on the virtual machine. Based on the info stated above, the participants of the case study, in context of TDSE, implemented a system improvement - a conditioned alarm email. It is automatically sent to the relevant stakeholders once CPU load on the relevant virtual machine reaches 60%. This countermeasure allows the test engineers responsible for nightly runs analysis save significant amount of time on debugging and contacting other domain experts.

Specified test cases revealed the lack of access rights for a big group of domain experts to particular assets. Therefore the test cases could be used in the future as a documentation on security aspects of the system in context of the TDSE approach.

At the very end the test engineer extended the related to the use case software engineering knowledge graph with the information regarding cause-effect relationships, links between cause and effect assets, equivalence classes for the correct and risky output, information about the possibilities of control and measurement, as well as the information about the levels of control over the relevant properties of the investigated assets. The model implemented in Neo4j [5] and depicted in Figure 5.7, the code is presented in Appendix A 8.2 of the paper.

5.5 Algorithm Performance – Evaluation

The research results have been evaluated in a case study with the relevant domain experts: quality manager, test engineer, test automaton engineer, software developer and leanOps engineer, based on the requirements specified in Section 4.1 during evaluation workshops. The traditional test case design and planning approach of the software engineering department of the Company A has been compared to the novel multi-aspect test case specification approach. The use case for automated test execution process concerning algorithm performance has been discussed, in order to better understand the process and needed knowledge that domain experts refer to when specifying test cases. Execution of automated tests on a remote virtual machine using Jenkins[6] has been discussed, as well as the important links between hardware and software parts of the system. The multi-disciplinary domain knowledge required for test case design and represented by the software engineering network has been investigated, in order for domain experts to be able to identify an efficient set of tests for the optimal requirements coverage. The domain experts typically rely on implicit knowledge which is typically limited by a single discipline, therefore the traditional method includes mental integration of single discipline-specific models by the domain experts.

[5] Neo4j tool: `https://neo4j.com/`
[6] Jenkins: `https://www.jenkins.io/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.5.1 Multi-aspect test case specification meta-model and model requirements

The first part of the evaluation is evaluation of the multi-aspect test case specification meta-model and model against the requirements (Section 4.1). Following requirements have been elicited from the literature and preliminary workshops with the domain experts:

R1.1 Representation of cause-effect relationships and testing specific visual elements and links between them. In the traditional approach, the domain experts were not familiar with the standard FMEA procedure [Stamatis, 2019], however, they conducted similar to FMEA steps in order to identify potential problems or the causes of the existing problems. Whenever there was a potential problem or an improvement, they organized brainstorming sessions where they viewed their engineering graphs from single disciplines and discussed future steps. Therefore MATCS implicitly introduced FMEA to the stakeholders and allowed them to follow the path from cause to the effect on a unified multi-disciplinary knowledge graph. Moreover, it was the first attempt to looking at the software system from the CPPS point of view. The domain experts liked the idea very much. Following, the domain experts stated that the implementation of the representation of cause-effect relationships and testing specific visual elements and links between them fulfills the requirement very well.

R1.2 Representation of testing knowledge. Traditional approach allowed the domain experts to store test case related knowledge in a test management system, using various articles with different levels of access to information. This knowledge, however, has not been bound to a knowledge model, and is, therefore, not well organized. In contrast, MATCS supports sharing and extension of testing knowledge, such as information regarding the equivalence classes for different effects, levels of measurement and control, in a unified multi-disciplinary knowledge model. The domain experts found the implementation of representation of testing knowledge in MATCS very systematic, while in the traditional approach it is not systematically represented. However, current capabilities of the Neo4j tool do not allow to represent the equivalence classes the same way, e.g. sometimes the name of the equivalence class displayed, and sometimes - the range of values. This is a technical limitation of the tool.

R1.3 Knowledge querying and iterative knowledge extension. In the traditional test related knowledge specification approach related to multi-disciplinary testing, there was no systematic way of making implicit expert knowledge explicit. The domain specific knowledge model allowed the domain experts for systematic knowledge collection in and from a knowledge model. Knowledge querying and iterative knowledge extension was not present in the traditional approach, while feedback from the domain experts shows that the implementation of this feature fulfills the requirement very well.

5.5.2 Multi-aspect test case specification method requirements

The second part of the evaluation is evaluation of the multi-aspect test case specification method against the requirements (Section 4.1). Following requirements have been elicited

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

from the literature and preliminary workshops with the domain experts:

R2.1 Key concepts. Key concepts including both, the knowledge model and the method concepts have been presented to the domain experts in Table 4.2. In the traditional approach such aspects have been presented as legends for the graphs or methods, therefore it was not possible to combine testing and system key concepts. The representation of the key concepts in Table 4.2 fulfills the requirement for the key concepts well.

R2.2 Risk assessment and prioritization. Multi-disciplinary risk assessment was conducted in a form of brainstorming in the old approach since there was no underlying multi-disciplinary knowledge model. It was not possible to store the structured outputs related to the system risks from different departments/disciplines in a systematic way. Extended multi-view risk assessment based on [Biffl et al., 2021] allowed the domain experts to conduct iterative multi-disciplinary risk assessment in a systematic way. The domain experts rate extended CPPS-RA approach higher than the traditional one.

R2.3 Equivalence classes concept support. This requirement was a part of the "Multi-aspect testing case specification" during the very initial requirements elicitation phase. However, this requirement needs a separate discussion. The idea of using equivalence class partitioning [Burnstein, 2003] of the test data space has been taken from the software development domain, where it is already often in use. Therefore the domain experts found the technique supportive for both of the presented traditional and novel approaches. The domain experts defined the minimum possible set of prioritized test cases thanks to the application of equivalence class partitioning technique and risk regions definition from the risk assessment step.

R2.4 Multi-aspect test case specification. Specification of test cases par of multi-aspect test case specification method by itself is a set of logically interconnected well established testing techniques. However, the techniques are applicable for single disciplines only. The domain specific knowledge models make the multi-aspect test case specification usable across multiple disciplines. Taking into consideration the underlying CPPS-RA, PPR, FMEA concepts, the domain experts rated the multi-aspect test case specification method as a method very well fulfilling the multi-aspect testing requirement, while the traditional approach was not able to support the domain experts in multi-aspect test case specification.

R2.5 Support for multi-aspect test automation. Similar to the multi-aspect test case specification requirement, support for multi-aspect test automation is based on multiple other concepts, such as CPPS-RA, PPR, FMEA which makes it different from the traditional software engineering test automation approaches. It is more powerful in multi-disciplinary environments, which the domain experts have highlighted. The traditional approach was not able to support the domain experts in multi-aspect test automation. MATCS allows the domain experts to establish the scope and limitations for the tests to automate. However, it is still not easy to implement automated test cases due to lack of access to various assets by the experts.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.5.3 Comparative analysis of traditional and MATCS approaches

Knowledge representation requirements	Tradit. approach	MATCS approach
R1.1 Representation of cause-effect hypotheses	- -	++
R1.2 Representation of testing knowledge	-	+
R1.3 Knowledge querying and iterative extension	- -	++

Table 5.10: Company A - Analysis of traditional testing knowledge representation and the MATCS approach.

Test case specification requirements	Traditional approach	MATCS approach
R2.1 Key concepts	-	+
R2.2 Equivalence classes concept support	++	++
R2.3 Risk assessment and prioritization	+	++
R2.4 Multi-aspect test case specification	-	++
R2.5 Multi-aspect test automation	- -	O

Table 5.11: Company A - Analysis of traditional test case specification and the MATCS approach.

In Tables 5.10 and 5.11, columns represent the traditional and novel test case specification approaches. While the rows of the Table 5.10 represents knowledge representation related requirements, Table 5.11 represents the method requirements. The cells of the Tables 5.10 and 5.11 display the values based on a 5-point Likert scale [Robinson, 2014] which represent the evaluation results. The signs + (++) indicate the discussed approach to satisfy the requirements well (very well), O indicates partial fulfillment, and - (- -) indicate low (very low) fulfillment of the requirement by the test specification approach.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 6

CPPS Engineering Use Case: Aluminium Surface Cleaning Process

This chapter introduces the illustrative use case from the large-scale *Company B - a global expert in aluminium manufacturing*. The company employs more than 2,500 employees in 8 countries worldwide and delivers aluminium products to customers from mobility, construction industry, packaging, electrical and mechanical engineering, medical technology, as well as consumer goods sectors. One of the plants of Company B is located in Austria and is dedicated to aluminium casting, direct extrusion and impact extrusion of aluminium products. Around 500 employees contribute to it. The CPPS infrastructure of the floor includes multiple robotic elements responsible for laser and MIG welding, washing process of the aluminium products in order to prepare them for MIG welding or the customer, as well as quality control of the products. Examples of the final products are battery infrastructure, pressure plates, chassis components, windows, swimming pool roofs, generators, motors etc. For the purposes of the thesis, together with the domain experts from the Company B, use case *"Aluminium Surface Cleaning Process"* has been selected.

Section 6.1 is designed to present the detailed use case analysis. In Section 6.2 the process of building PAN is described. Section 6.3 presents the multi-view risk assessment conducted in cooperation with the domain experts. Section 6.4 addresses the multi-aspect test case specification process within the use case. Finally, Section 6.5 presents the evaluation of the method by the domain experts from the Company B.

6.1 Aluminium Surface Cleaning Process – Use Case Analysis

In order to prepare aluminium details for further welding or packaging, aluminum surface cleaning is required. Aluminum surface cleaning within the company production process is done using an industrial washing machine (Figure 6.1).

Figure 6.1: Company B - Industrial washing machine for aluminum surface cleaning.

The aluminium details are delivered by a transport robot to the machine one-by-one and undergo degreasing, etching, washing and drying procedures in chambers one-six in first in - first out order. Once the detailed has been cleaned, another transport robot picks it up from the other side of the machine and deliveries it to the next station. For the process distilled water is used. The process is depicted in Figure 6.2. The washing machine consists of six chambers designed for different purposes corresponding to the aluminium surface cleaning process steps:

- Step 1: Degreasing. Chamber one. During this step degreasing of the aluminium products takes place. This is done in order to be able to remove the oily layer which appears on the product during the stamping process due to the machines lubrication.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

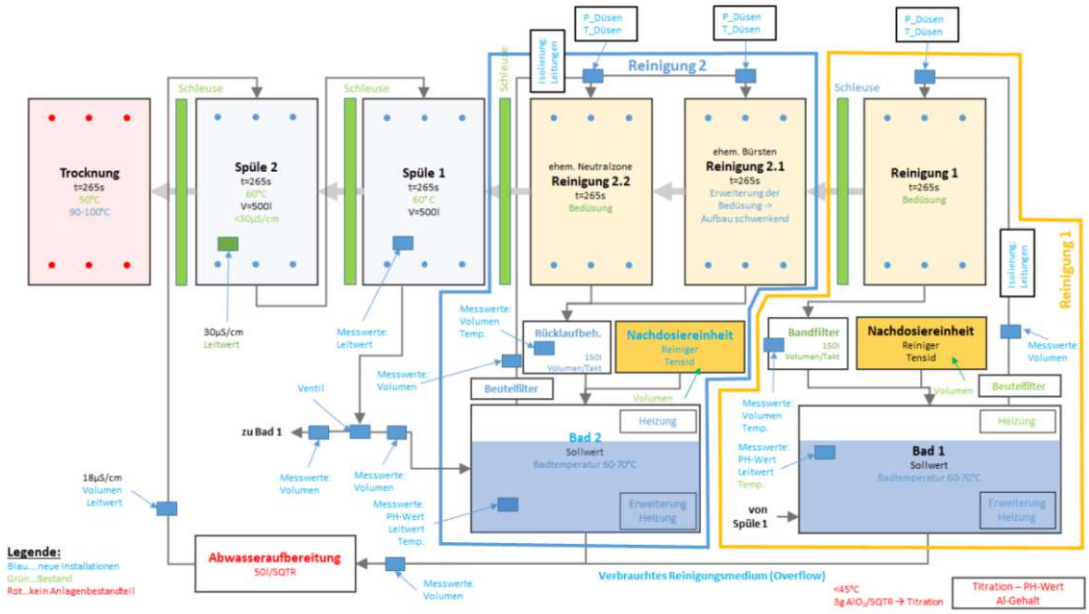

Figure 6.2: Company B - The five steps industrial aluminium surface cleaning process. Based on a Company B report file.

This is achieved by spraying of a mixture of an industrial alkaline cleaner with a tenside solution and distilled water on the aluminium surface. Chemistry tank one is used in step one.

- Step 2.1 and Step 2.2: Micro etching. Chambers two and three combined. In this chamber the chemistry tank two is used. In order to prepare the aluminium product for further welding, etching process takes place.
- Step 3: First cycle of rinsing with water. Chamber four. Once the aluminium part has reached step three, washing with clean water takes place. At this stage the mixture of the alkaline cleaner with and tenside solution is removed from the aluminium surface. Water from this step flows to the water tank one in order to be re-used.
- Step 4: Second cycle of rinsing with water. Chamber five. One more cycle of cleaning is done during step five, in order to remove the remaining grease and physical particles from the surface.
- Step 5: Drying. Chamber six. Step five is designed in order to finalize the cleaning process. In chamber six the aluminium part undergoes drying process.

Chambers one, two and three are designed to remove the oxidation level using a liquid alkaline cleaner on the aluminium in order to prepare it for the next steps. Step 1 of the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Figure 6.3: Company B - Spraying nozzles and the transport system in chamber one.

aluminium surface cleaning process (Figure 6.2) is done in chamber one, steps 2.1 and step 2.2 happen in chambers two and three. For the chambers one two and three the following is valid: the best washing effect can be achieved with pH=10,4 for the chemical bath. In order to achieve the pH level, water temperature in the chamber should be in the range of 60-70°C. A decrease of the water temperature leads to an increase of the pH level. If pH<10,4, brown strains displayed on Figure 6.4 occur. In this case the aluminium products needs to be melted and produced again. The conductivity of the bath is defined by the amount of dissolved aluminium in the water. It is not known in advance how much aluminium will be dissolved by the chemistry from each aluminium part. The maximum amount of dissolved aluminium in these wash tank should remain under 300mg/L. For an increased amount of the dissolved aluminium in the chambers an increased amount of the washing chemistry is required in order to keep the appropriate pH level. Titration test[1] is conducted in order to measure the aluminium and the chemistry concentration in the bath and keep the appropriate pH level by adjusting the amount of chemistry. A titration test is an experiment where a volume of a solution of known concentration is added to a volume of another solution in order to determine its concentration. The test is supported by the formula:

6.1(A-B), where:*
A is 10mL of the base solution with 100mL of fully desalted water
B is alloy concentration in liquid

[1]Titration test: `https://chem.libretexts.org/Bookshelves/Introductory_Chemistry/Introductory_Chemistry_(CK-12)/21%3A_Acids_and_Bases/21.17%3A_Titration_Experiment#:~:text=A%20titration%20is%20an%20experiment,titrations%20can%20also%20be%20performed.`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Figure 6.4: Company B - Aluminium brown strains coloration defect.

Chamber four is required for step 3 of the aluminium surface cleaning process. In this chamber the chemistry on the aluminium surface is rinsed by desalted water. The conductivity of water in the fourth chamber could be $<1000\mu S/m^2$. After the third chamber the aluminium details are shifted to the fourth chamber for step 4 of the process. There the details undergo another cycle of rinsing. At this step the conductivity of water usually remains $< 30\mu S/m^2$.

The focus of the use case is aluminium surface cleaning cycle time improvement by decreasing the degreasing time in chamber one. Washing cycle time is the sum of washing times from each individual chamber. Based on the domain experts' opinions, improvement could be achieved by changing parameters of the wash tank related to chamber one. The goal of MATCS is to reveal the cause-effect paths, build hypotheses based on the output of extended CPPS-RA, specify test cases to validate the hypotheses stated by the domain experts, e.g. whether the washing cycle time could be improved by changing parameters of the wash tank related to chamber one, and define the automation scope and implement the skeleton for the possible future test cases in Gherkin notation.

Project stakeholders

Figure 6.5 illustrates the stakeholders involved in the aluminium surface cleaning project

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

project, their interactions and dependencies. Stakeholders are represented by person-like icons of different colors. The arrows corresponding to the color of a stakeholder represent information requested by the stakeholder from the others. Incoming arrows represent the response from the others. In order for a quality engineer to find out whether aluminium surface cleaning time could be improved, knowledge from other stakeholders is necessary. The parameters to investigate are coming from a software driven machine and involve chemical aspects. Therefore, at the Company B, the quality engineer needs to ask for information system engineers, mechanical and chemical engineers, the washing expert and the production leader about various influences which could support the improvement.

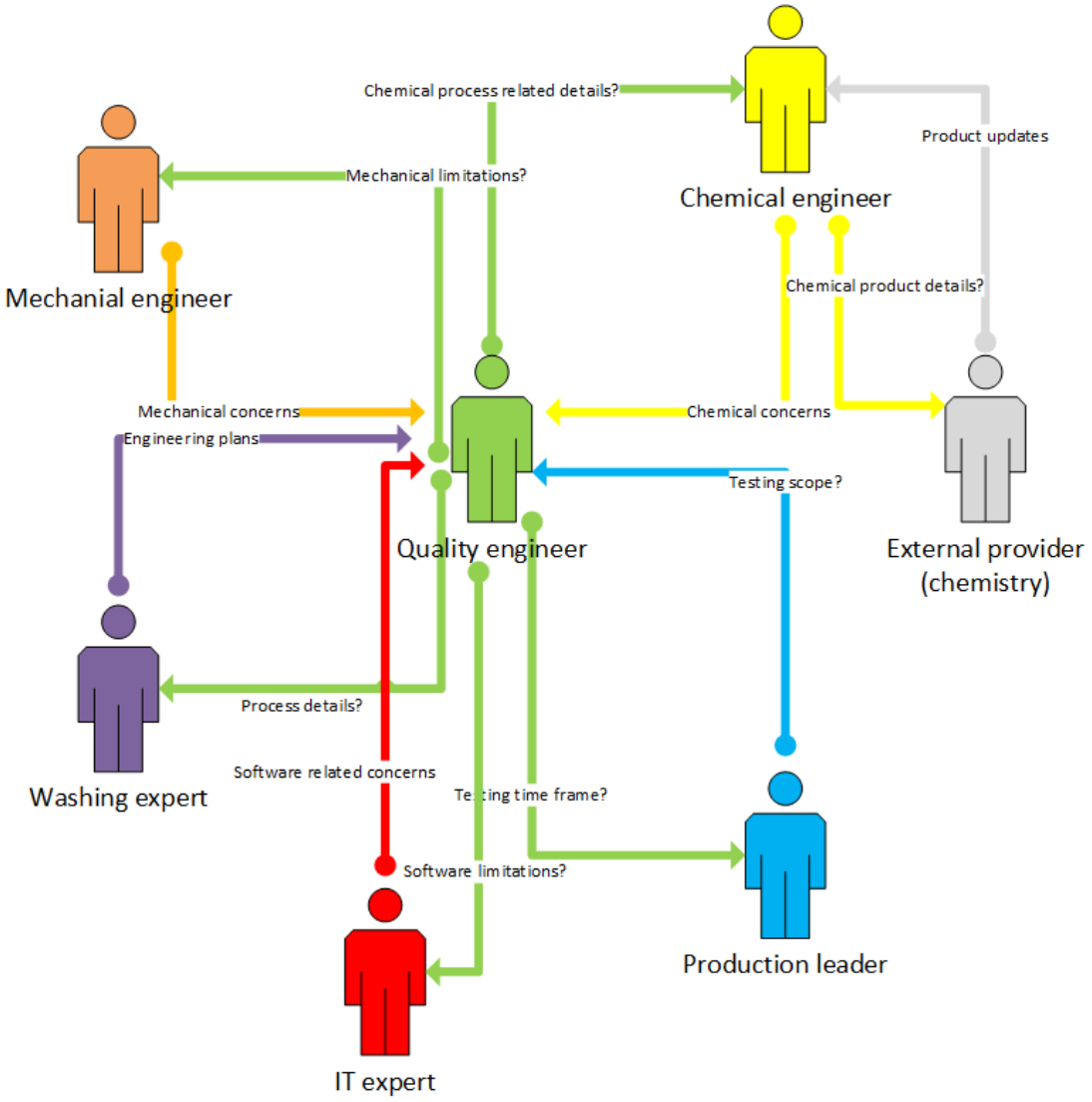

Figure 6.5: Company B - Aluminium surface cleaning project stakeholders.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

As-Is-Analysis: Test case specification without MATCS method.

In order to prevent future production issues or introduce improvements, the domain experts from Company B build a multi-disciplinary PPR [Winkler et al., 2021] knowledge graph and then follow CPPS-RA [Biffl et al., 2021] steps. At the brainstorming stage they identify the 5-M-Method or, in other words, Ishikawa diagram[2], which helps them to identify the factors which could influence the end effect. They draw the effect and provide links to: "Man-Power", "Method", "Milieu" "Matter" and "Means" types of assets. Following the experts discuss their assumptions and come to a shred conclusion regarding the matter of the source of the issue or improvement. Most of the sources lay within the company scope. However, some of the sources might come from the third party providers. The company has a contractual bound with the providers regrading the support and consultancy. Once it is clear from which "M" the issue/improvement can come from, the domain experts follow the FMEA steps based on the PPR knowledge model. Once the possible causes of a potential issue or improvement have been identified, a test case is specified and documented on paper. Value of the test case parameters are usually single values. The goal of such test case is reaching the target reproducibility of the potential issue or the improvement. Often it is not easy to execute the test case at all, since the production line should not be stopped for such matters without an approval from the customers or the production leader, change of the parameters at production stage can be very risky and the simulation for it often does not exist. In case it is possible to execute such a test case, it has to be approved by the production leader. After the execution of the test case the confirmed values are either applied, applied with modifications of parameters of the other related production units, or rejected. The test case and the results are then stored digitally in a report in a shared folder. Such test cases do not run on a regular basis and are not automated. Test case automation has not been considered yet.

6.2 Aluminium Surface Cleaning Process – PPR Asset Network Definition

This section aims to give an overview of the PAN implemented by the domain experts from the use case description. The PAN consists of four input and one output products, one process step and sixteen resource assets, including the OPC UA controller. The domain experts have identified five types of links representing relationships between the network assets: Product-Process links, Process-Resource links, Mechanical and Electrical links, as well as Controller-Resource link. The steps for building a PAN are described in Section 2.1.1 and depicted in Figure 4.2. The role of the PAN in test case specification is displayed in a more generic Figure 6.6. Figure 5.3 displays the software engineering PAN representing the use case relevant set of products, processes and resources, built by the domain experts based mostly on their professional knowledge and engineering plans. The domain experts shared their knowledge, which was later depicted in the graph using

[2]Ishikawa diagram: `https://link.springer.com/book/9789401176903`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

UMLet[3] tool. Following, the knowledge model has been implemented in Neo4j software[4]. The code of the final knowledge model can be found in Appendix B 8.2.

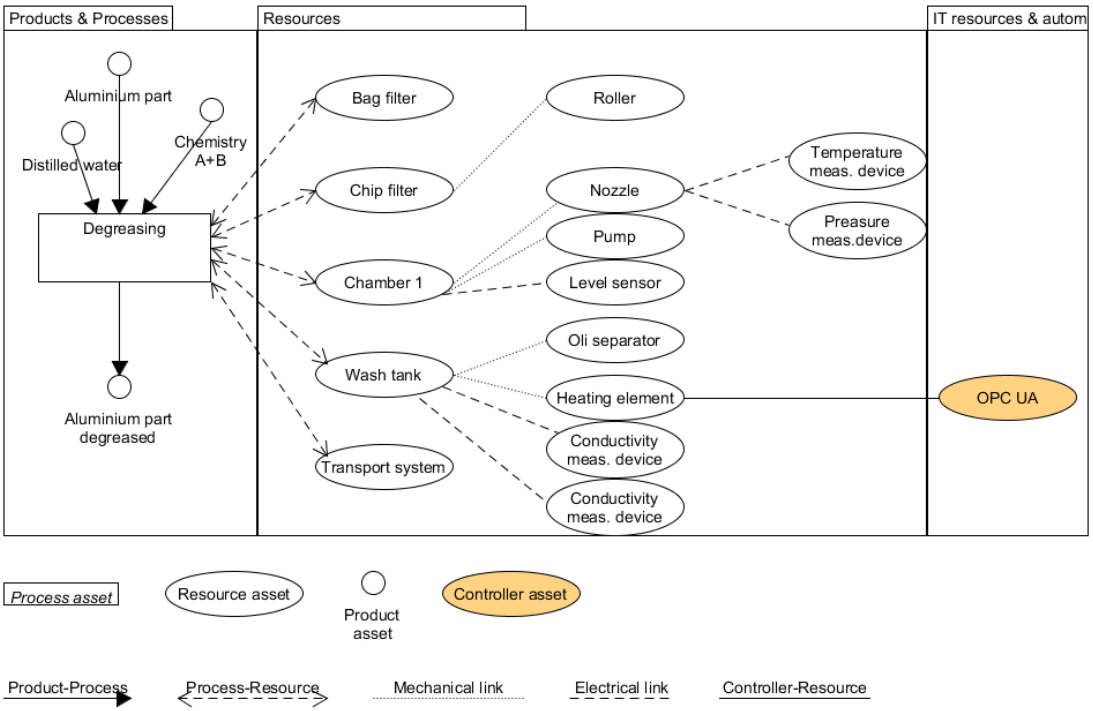

Figure 6.6: Company B - CPPS engineering PPR Asset Network (PAN).

6.3 Aluminium Surface Cleaning Process – CPPS Multi-View Risk Assessment

This section illustrates how domain experts, such as quality engineer, chemical engineer and cleaning expert, built *CPPS Engineering Network (CEN)* based on the CPPS-RA meta-model [Biffl et al., 2021]. Following the experts identified related aspects in the CEN which potentially could contribute to the aluminium surface cleaning cycle time improvement. In order to do so they analyzed product, process, resources (as building blocks of the model), and extended the basic model with causes, effects, and dependencies. The extended CEN consists of effects to be observed, related production processes and system resources, influenced by a set of root causes (details in the legend of Figure 6.8).

The multi-aspect risk assessment Process was conducted in five main steps with regards to the multi-view risk assessment method [Biffl et al., 2021] and the extra steps described in 4.4.

[3]UMLet tool: `https://www.umlet.com/`
[4]Neo4j tool: `https://neo4j.com/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Step 2.1. Identify risk drivers. Within this step of the multi-view risk assessment process, Table 4.4 has been presented to the domain experts. Based on the nature of the specified risk and own expertise, the domain experts came to a conclusion that the presented system requirement is performance, and therefore the risk driver is *Performance risk driver*.

Step 2.2. Identify risk regions. The decrease of the aluminium surface cleaning cycle time is a system improvement. In this scenario, the goal is to identify the assets and their properties which lead to the improvement (desired effect). Therefore two risk regions are considered: default region with values when the system is expected to work as designed and the improvement region - the region which should be confirmed. The regions are represented by equivalence classes during the test case specification process.

Step 2.3. FMEA: Identify risk and informal cause candidates. In this step, domain experts followed the FMEA process with the goal and scope specified in order to define the SuI. The task of this step was to identify and prioritize candidate improvement modes, effects, and risks. For a selected effect E, the domain experts built on CPPS engineering knowledge in the project to elicit a list of possible cause and hypothesis candidates. The stakeholders used notation such as "Effect E_{11} could potentially be caused by cause C_1, C_2, C_3 or their combinations." This means that for the desired risk *aluminium surface cleaning cycle time decreased*, the domain experts identified as possible cause candidates: temperature of the water in the first washing tank, or freshness of the content of the washing tank which would allow increase concentration of the detergent, or decreased aluminium concentration in the water, or free from oil input products.

Step 2.4. Multi-view risk assessment with CPPS engineering network. Step 2.4 includes 3 important sub-steps: 1) explore software engineering network; 2) analyze cause candidates and cause-effect pathways; 3) build hypothesis linked to the CEN.

Step 2.4.1. Explore SuI engineering network. In this step, the domain experts identified assets which are relevant to represent informal cause candidates. The domain experts started in the CEN from the asset which represents the SuI, e.g., the improvement. The following query helps to fetch all assets linked to the SuI:

```
MATCH (n:Product:PPRAsset {name:"Degreasing"})-[*]-(m)
RETURN n,m
```

Following, they explored CEN assets in an iterative way. The exploration happened by following selected links between assets, e.g., mechanical or electrical links, which could be related to the selected effect. The domain experts identified five pathways:

1. S_0 (Degreasing) -> S_1 (Wash tank),
2. S_0 (Degreasing) -> S_1 (Wash tank) -> S_2 (Heating element),
3. S_0 (Degreasing) -> S_1 (Wash tank freshness) -> S_2 (Wash tank pH level),

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4. S_0 (Degreasing) -> S_1 (Amount of grease on the aluminium part),
5. S_0 (Degreasing) -> S_1 (Concentration of alloy in the wash tank).

Step 2.4.2. Analyze cause candidates and cause-effect pathways. The domain experts identified cause candidates linked to the CEN elements found in the risk assessment Step 2.1, such as temperature of the water in the first washing tank, or freshness of the content of the washing tank which would allow increase concentration of the detergent, or decreased aluminium concentration in the water, or free from oil input products. A cause-effect diagram (Figure 6.8) links the SuI, e.g. aluminium surface cleaning cycle time decrease, to technically connected assets as foundation for defining a cause and a pathway that connects the root cause to the SuI and the improvement effect. All four cause-effect paths appeared to be reasonable. In order to identify the most relevant path, the domain experts prioritized the cause candidates. After this step, only one cause-effect path remained: S_0 (Degreasing) -> S_1 (Wash tank) -> S_2 (Heating element).

Step 2.4.3. Build hypothesis linked to the CEN. The domain experts used a simple restricted language to express their hypotheses based on cause candidates linked to CEN elements, e.g., $H(E_{11}; C_1)$, where E_{11} represents the effect aluminium surface cleaning cycle time decreased, and C_1 water temperature increase in the first wash tank.

6.4 Aluminium Surface Cleaning Process – MATCS Application

This section illustrates how the quality engineer from the Company B, together with the domain experts applied the multi-aspect test case specification method in order to generate test cases based on the multi-aspect cause-effect knowledge graph prepared by the domain experts. Following, the test activities were planned, executed and evaluated. The process includes five steps according to the definition of MATCS method, which include: 1) scoping; 2) test case derivation; 3) test control and measurement; 4) test automation; 5) test data collection and analysis.

6.4.1 Step 3.1. Scoping

As described by the MATCS method, the input parameters for this step are: the cause-effect graph with identified possible causes for specified effects, hypotheses built based on the cause-effect relationships and multi-aspect environment experience of the domain experts. At this step, the domain experts created a document where they described the desired and undesired effects, the assets linked to the defects, and the possible causes contributing to the effect.

Current effect, e.g. standard aluminium surface cleaning cycle time, happens when the system works as expected, e.g. all the assets, contributing to the effect, correspond to the product requirements.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Cause	EC_1 OK	EC_2 Impr.	Can measure	Can control	Levels of control	Exec
$C_1 : Watertemperatureincreased$						

Table 6.1: Company B - Template table for cause candidate EC values

TC N	C_1	Exp. result	Act.result	H_n	Can control	Can automate	Level of exec.
TC 1							
TC 2							

Table 6.2: Company B - Template table for test cases.

The desired effect, e.g. aluminium surface cleaning cycle time improved, takes place when the identified cause candidate is the cause responsible for the change and it can be confirmed. Otherwise, other cause-effect paths need to be taken into consideration. The tester, together with the domain experts, built two hypotheses based on the above stated information:

Null Hypothesis: $H_0(E_{11}; NOR(C_1)$. This means that the temperature increase in the wash tank does not lead to the aluminium surface cleaning cycle time improvement. Another representation: P(duration) != f(t).

None-Null hypothesis: $H_1(E_{11}; OR(C_1))$. This means that the identified cause candidate is an actual cause of the improvement. Another representation: P(duration) = f(t).

Following the domain experts went iteratively through the cause candidate assets from the hypotheses and prepared a draft Table 6.1 for the cause candidate level values. C_n represents the number of the cause candidate from the knowledge graph, EC_n represents the number of the equivalence class. The experts defined two possible cause candidate levels per asset, therefore the Test case template Table 6.1 contains two rows. This means that each of the three asset levels has two equivalence classes: the cause variable could either have the specification (EC_1 OK) value or risky value (EC_2 Improvement). The template table is illustrated by Table 6.2.

6.4.2 Step 3.2. Test case derivation

In this step the domain experts and the quality engineer filled in the values for the cause candidate and the equivalence classes according to their experience, logs and technical requirements. The assumption is that the water temperature in the first wash tank caused by C_1, e.g. raising the temperature of using the heating element related controller user interface. It is known by the domain experts that, in order to keep the pH level = 10.2, the water temperature range in the first wash tank should be 60-70°C. Before the experiment the value for the operator controller was set to max 65°C. From the past observations, the domain experts knew that the water could be overheated by 1°C. Figure 6.7 displays the controller user interface: the washing chambers with their current parameters. The temperature is displayed in the right lower corner of the Figure.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

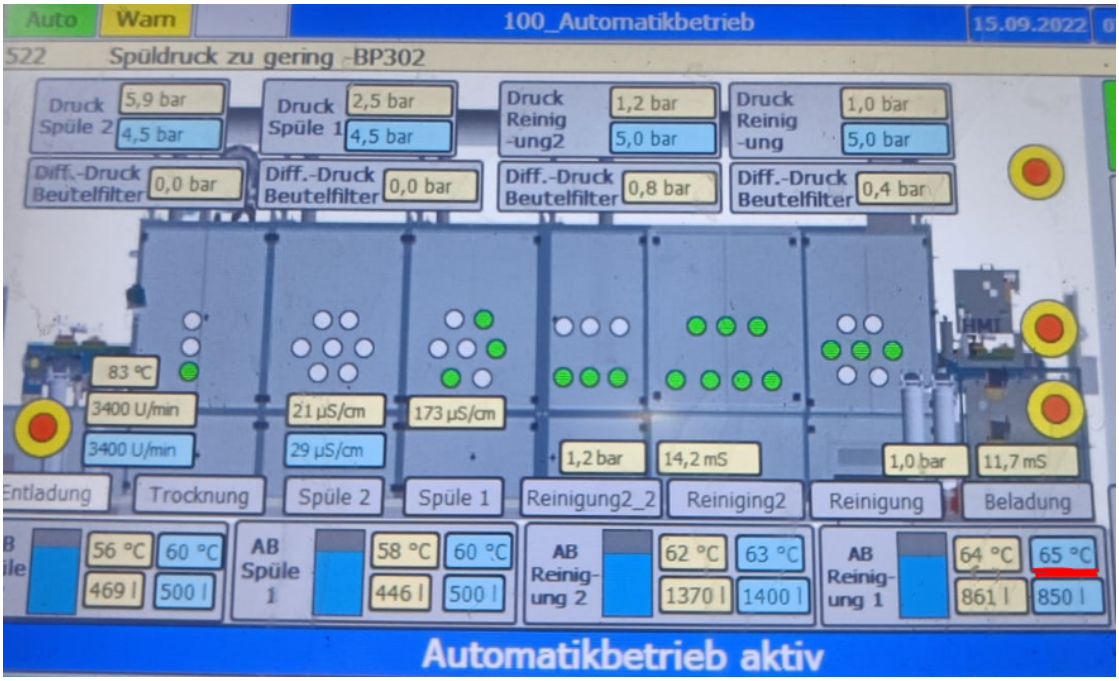

Figure 6.7: Company B - Temperature setting before the experiment.

At first, the domain experts suggested to increase water temperature to 70°C and collect the data from the 20 runs. After each run one aluminum part is considered clean. The domain experts modified the parameter value from the operator user interface during the maintenance mode and started the system with the new value. However, the system overheated and the temperature reached 71°C within two runs. According to the user's manual of the controller, 71°C is the maximum possible water temperature for the first wash tank and in case of overheating, an alarm should be triggered. Moreover, the software logic does not allow to input a number higher than 71 to the input field. This lead the domain experts to the conclusion that the improvement equivalence class should be 67-69, taking into consideration the overheating factor.

For the second iteration of the experiment, the domain experts adjusted the water temperature to max 69°C without stopping the production line. After the 20 runs, the quality engineer together with the chemical engineer compared the past average washing time (225sec) with the new average washing time (201sec, the data can be found in the AppendixB 8.2). The aluminium surface cleaning cycle time improved by 10%. Following, the quality engineer filled Table 6.3 with the equivalence classes from their findings:

Based on the analysis, the experts could fill in the table 6.3 with the corresponding values. The first test case is the default system behavior test case, which means that if all the values are set correctly, the improvement will not happen. The second test case is the improvement test case, meaning that if the water temperature is the range [67-69°C], the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Cause	EC_1 OK	EC_2 Impr.	EC_3 Crit.	Can measure	Can control	Levels of control	Exec
$C_1 : Watertincreased$	[60-66°C]	[67-69°C]	[70-71°C]				

Table 6.3: Company B - Table for the cause candidate with equivalence classes values after experimentation.

washing time decreases. The third test case is the test case for the critical temperature value. An FMEA countermeasure was introduced and tested in the past. The third test case should only be executed when security alarms are tested. Following, the quality engineer updated the test case Table 6.4.

TC N	C_1	Exp. res.	Act. res.	H_n	Can control	Can automate	Exec
TC 1	[60-66°C]	Time avg 225 sec		H_0			
TC 2	[67-69°C]	Time avg 201 sec		H_1			
TC 3	[70-71°C]	Alarm		H_1			

Table 6.4: Company B - Table for test cases with equivalence classes values.

6.4.3 Step 3.3. Test control and measurement

At this step, the test engineer in cooperation with the domain experts iterated through the set of test cases and defined the level of measurement and control. The following levels of control have been identified: all present domain experts had the access to the controller and could modify the temperature using its display.

The following relevant to the cause asset properties levels of measurement have been identified by the domain experts: all present domain experts could read the temperature values from the operator display and the logs software.

Following, the quality engineer added the info related to the degree of measurement and control to the cause (Table 6.5). Based on the results of the measurement and control analysis, the quality engineer defined which of the test cases could be executed manually: TC 1 and TC2 without concerns, and TC3 in order to test the safety aspect, Table 6.6.

Cause	EC_1 OK	EC_2 Impr.	EC_3 Crit.	Can measure	Can control	Levels of control	Exec
$C_1 : Watertincreased$	[60-66°C]	[67-69°C]	[70-71°C]	yes, auto	yes, auto	all DE	prod

Table 6.5: Company B - Properties of cause candidates

TC N	C_1	Exp. result	Act. result	H_n	Can control	Can automate	Exec.
TC 1	[60-66°C]	Time avg 225 sec		H0	yes	yes	prod
TC 2	[67-69°C]	Time avg 201 sec		H1	yes	yes	prod
TC 3	[70-71°C]	Alarm	Alarm	H1	yes	yes	prod

Table 6.6: Company B - Test cases.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

6.4.4 Step 3.4. Test automation

Based on the analysis from the previous steps and the discussion of the method and the automation possibilities, the quality engineer decided that running the specified test cases on a regular basis would make sense. However, in order to do so, software engineers which were not a part of the case study need to be involved. Moreover, it needs to be plan carefully far in advance due to the production line schedules, test and results monitoring. Therefore a Gherkin[5] scripts has been prepared for the Company B in order to support the software engineers in the future:

```
Feature: H0 testing
As a quality engineer,
I want to test time OK
So I can confirm time OK scenario

Given I assume "temperature increased" is the cause
And I apply temperature max "65"C
And I produce "20" parts
And washing time is avg "225"s
And quality is "OK"
Then I confirm OK scenario

Feature: H1 testing
As a quality engineer,
I want to test time improved
So I can confirm time improved scenario

Given I assume "temperature increased" is the cause
And I apply temperature max "69"C
And I produce "20" parts
And washing time is avg "201"s
And quality is "OK"
Then I confirm time improved scenario
```

For the potential case of a need of slowing down or speeding up the washing process, the test cases are stored in the domain knowledge model (Figure 5.7, Appendix B 8.2).

6.4.5 Step 3.5. Test data collection and analysis

In the final step, the quality engineer executed the first two test cases and validated the expected results against actual results. Table 6.7 represents the results of the test runs. In both cases, 20 aluminium parts were cleaned. None of the pieces came out of the

[5]Gherkin syntax: https://cucumber.io/docs/gherkin/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

TC N	C_1	Exp. result	Act. result	H_n	Can control	Can automate	Exec.
TC 1	[60-66°C]	Time avg 225 sec	Time avg 225 sec	H0	yes	yes	prod
TC 2	[67-69°C]	Time avg 201 sec	Time avg 201 sec	H1	yes	yes	prod
TC 3	[70-71°C]	Alarm	Alarm	H1	yes	yes	prod

Table 6.7: Company B - Test case results.

washing machine with a defect. The aluminium surface cleaning cycle time was improved by 10% by increasing the water temperature in the was tank one. This change can be introduced as a permanent solution at the Company B without risks and further process adjustments, e.g. the washing step can run faster without interrupting the production cycle already now.

After the test runs, the quality engineer could reject H_0 and failed to reject H_1. The investigated cause allows the improvement. Specified test cases revealed two dependent on each other parameters of the system and three sets of values for the parameters, where all of them have been confirmed by the quality expert. Therefore they could be used in the future as a part of the documentation on washing process in context of the TDSE approach.

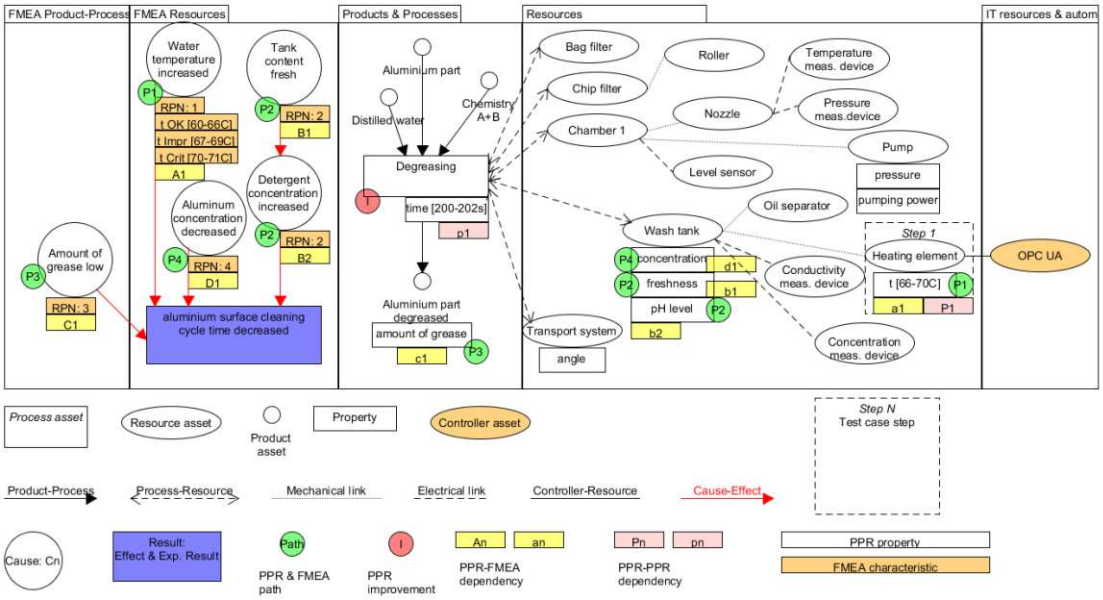

Figure 6.8: Company B - FMEA + PPR: cause-effect pathway (red arrows) leading from causes (white circles) to an FMEA effect (violet box) via linked assets in a SEN (green circles), including the test case steps (dashed rectangles).

At the very end the quality engineer extended the related to the use case software engineering knowledge graph with the information regarding cause-effect relationships, links between cause and effect assets, equivalence classes for the correct and improved

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

output, information about the possibilities of control and measurement, as well as the information about the levels of control over the relevant properties of the investigated assets. The model implemented in Neo4j [6] and depicted in Figure 5.7, the code is presented in Appendix B 8.2 of the paper.

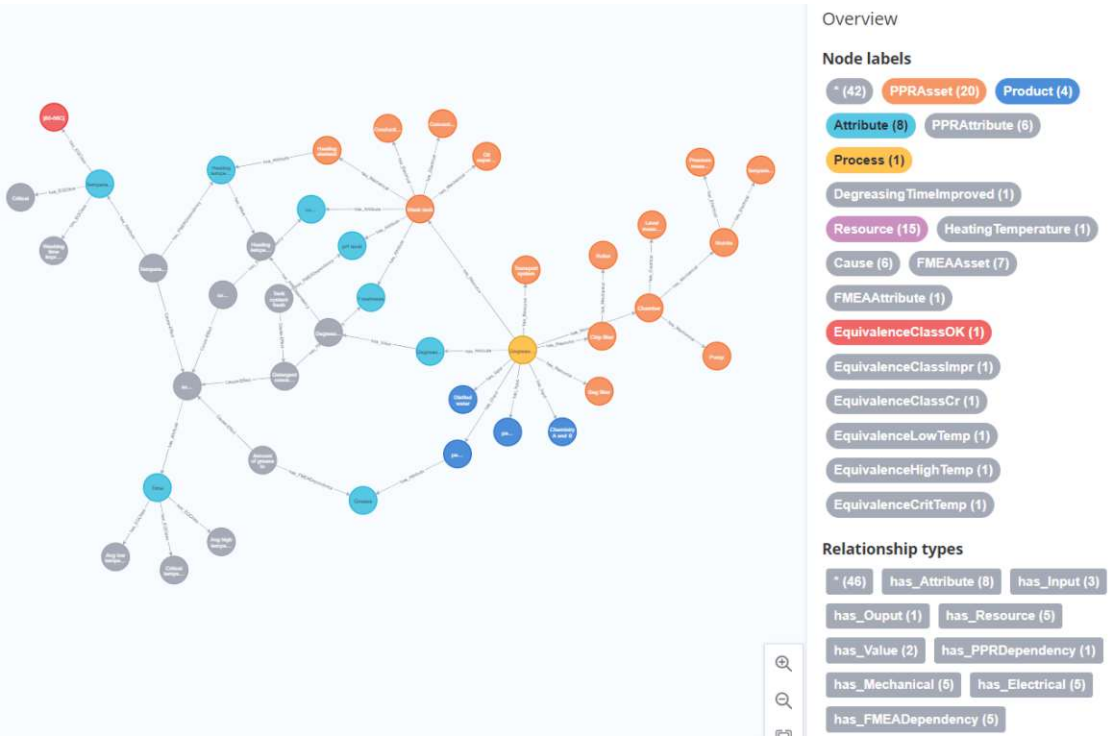

Figure 6.9: Company B: FMEA + PPR + test related knowledge in Neo4j: cause-effect pathway (link cause-effect on the left) leading from causes (gray circles on the left) to an FMEA effect (gray bottom circle on the left) via linked assets in a CEN (orange circles). Equivalence classes are linked to the FMEA attributes of the FMEA assets (circles linked to the blue circles).

6.5 Aluminium Surface Cleaning Process – Evaluation

The research results have been evaluated in a case study with the relevant domain experts: quality engineer, chemical engineer and washing expert based on the requirements specified in Section 4.1 during evaluation workshops. The traditional test case design and planning approach of the Company B has been compared to the "intermediate" PAN + CPPS-RA approach and the novel MATCS approach. The use case for industrial aluminum surface cleaning process has been discussed, in order to better understand the process and needed knowledge that domain experts refer to when specifying test cases. The multi-disciplinary

[6] Neo4j tool: `https://neo4j.com/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

domain knowledge required for test case design and represented by the CPPS engineering network has been investigated, in order for domain experts to be able to identify an efficient set of tests for the optimal requirements coverage. The domain experts typically rely on implicit knowledge which is typically limited by a single discipline, therefore the traditional method includes mental integration of single discipline-specific models by the domain experts.

6.5.1 Multi-aspect test case specification meta-model and model requirements

The first part of the evaluation is evaluation of the multi-aspect test case specification meta-model and model against the requirements (Section 4.1). Following requirements have been elicited from the literature and preliminary workshops with the domain experts:

R1.1 Representation of cause-effect relationships and testing specific visual elements and links between them. In the traditional approach of the Company B the domain experts used to rely on the knowledge graphs from single disciplines. The cause-effect relationships could not have been traced withing a single model. Therefore they were a part of the mental process of preparation for the risk assessment and testing of hypotheses. In the PAN + CPPS-RA approach representation of cause-effect relationships and links between them is supported very well (Figure 2.7). However, in the intermediate approach testing related knowledge is missing on the knowledge graph. Therefore the domain experts gave the partially fulfilment note to the intermediate approach. Representation of cause-effect relationships and testing specific visual elements and links between them in the MATCS approach is supported very well (Figure 4.6).

R1.2 Representation of testing knowledge. This requirement was elicited in order to allow the domain experts to store and retrieve hypothesis testing related knowledge, such as equivalence classes, level of control and measurement and levels of executing, in and from the knowledge graph. In both, traditional and intermediate approaches, the functionality is currently missing. In The domain experts could not say from the first view, which bubble from the MATCS model designed in Neo4j corresponds to which asset. This is the software limitation. Therefore MATCS approach supports the representation of testing knowledge well and further research on suitable software tools for knowledge models representation is needed.

R1.3 Knowledge querying and iterative knowledge extension. Knowledge sharing is very much related to knowledge querying and extension. The requirement stated that the domain experts from different disciplines should be able to store their knowledge in a unified knowledge model, should be able to retrieve and extend their knowledge and knowledge of their colleagues. The traditional approach did not support knowledge querying and iterative knowledge extension. Both PAN + CPPS-RA and MATCS approaches support such expert-model interactions very well (Figure 4.6).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

6.5.2 Multi-aspect test case specification method requirements

The second part of the evaluation is evaluation of the multi-aspect test case specification method against the requirements (Section 4.1). Following requirements have been elicited from the literature and preliminary workshops with the domain experts:

R2.1 Key concepts. The key concepts of the method, such as a table with the key terms descriptions, was not present in the traditional Company B approach. The domain experts used to search for the descriptions of the terms in numerous sources with documentation or ask the colleagues. In PAN + CPPS-RA the key concepts for PAN and CPPS-RA aspects are presented very well (Table 2.3). However, the key concepts related to the hypotheses testing are not described in the table. MATCS approach contains the easily readable and understandable key concepts Table 4.2 which allows the domain experts to understand the method without further help.

R2.2 Risk assessment and prioritization. The traditional approach included risk assessment and prioritization including the standard FMEA steps with the 5-M-Method represented in Ishikawa diagram based on brainstorming. The intermediate approach introduces the steps which allow the domain experts to consider well represented conditions from the underlying systems based on PAN. MATCS is built on the PAN and CPPS-RA approach and extended. The extension helps the domain experts to identify the system requirements which drive the risks and define the risk regions to minimize the test scope.

R2.3 Equivalence classes concept support. Equivalence class partitioning technique is well known in software engineering context. However, the domain experts from the CPPS domain of the Company B heard about it for the first time. Which means the the traditional approach did not include the support for it. PAN and CPPS-RA does not include it either, since it is not designed for hypotheses testing. MATCS includes support for equivalence classes concept. The domain experts defined the minimum possible set of prioritized test cases thanks to the application of equivalence class partitioning technique and risk regions definition from the risk assessment step. The domain experts found the technique very fitting to their traditional, intermediate and the new MATCS approach.

R2.4 Multi-aspect test case specification. Based on PAN and extended CPPS-RA, MATCS method gives clear directions regarding the multi-aspect test case specifications. In the traditional approach the steps were not defined, the knowledge graph did not exist or was not reusable and did not include all required system views and aspects.

R2.5 Support for multi-aspect test automation. Test automation has not been considered by the domain experts from the Company B. After MATCS has been introduced to them, the domain experts discussed the possibilities of automation of the specified test cases. It is very dangerous to let the tests run without supervision, therefore precise planning of the data source collection and storage, and the execution schedule are needed. Test case automaton scope, limitations and a possible course of actions have been documented and Gherkin scripts were implemented for the future use by the Company B. Therefore MATCS only partly fulfills the requirement.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

6.5.3 Comparative analysis of traditional, PAN + CPPS-RA and MATCS approaches

In Tables 6.8 and 6.9, columns represent the traditional and novel test case specification approaches, while the rows of the Table 6.8 represents knowledge representation related requirements, while Table 6.9 represents the method requirements. The cells of the Tables 6.8 and 6.9 display the values based on a 5-point Likert scale [Robinson, 2014] which represent the evaluation results. The signs + (++) indicate the risk assessment approach to satisfy the requirements well (very well), O indicates partial fulfillment, and - (- -) indicate low (very low) fulfillment of the requirement by the test specification approach.

Knowledge representation req.	Tradit.	PAN + CPPS-RA	MATCS
R1.1 Representation of cause-effect hypotheses	- -	O	++
R1.2 Representation of testing knowledge	-	-	+
R1.3 Knowledge querying and iterative extension	- -	++	++

Table 6.8: Company B - Analysis of traditional testing knowledge representation, PAN and CPPS-RA, and the MATCS approach.

Test case specification requirements	Tradit.	PAN + CPPS-RA	MATCS
R2.1 Key concepts	-	O	++
R2.2 Equivalence classes concept support	- -	- -	++
R2.3 Risk assessment and prioritization	+	++	++
R2.4 Multi-aspect test case specification	O	O	++
R2.5 Multi-aspect test automation	- -	- -	O

Table 6.9: Company B - Analysis of traditional test case specification, PAN and CPPS-RA, and the MATCS approach.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 7

Discussion and Limitations

At this point of time, all results have been evaluated and presented throughout the thesis. This chapter presents the comparative analysis of MATCS applied in software and CPPS engineering environments, discusses the Research Issues (RIs) stated in Section 3.1, as well as limitations of the solution approach. At first, Section 7.1 aims to present the comparative analysis of MATCS applied at Software and CPPS engineering environments. Section 7.2 is designed in order to revisit the RIs and discuss the output. At the end, in Section 7.3, limitations concerning data collection, applicability of the method and scope of the solution are discussed.

7.1 Comparative Analysis of Case Study Results

This section is designed to provide the comparative analysis of MATCS applied in software and CPPS engineering environments. Both, Company A and Company B are large-size companies with multi-disciplinary software-intensive system engineering nature. For the Company A with a software engineering department it is valid that the software system is distributed and the software experts are not located in the same country, while the production system of the Company B represents discreet engineering domain. The domain experts from the Company A were not familiar with the PAN and CPPS-RA concepts before the workshops, while for the domain experts from the Company B both concepts are a part of an ongoing pilot project, however, without the testing approach. Therefore at the Company A MATCS approach was evaluated against the traditional approach, while at the Company B the evaluation of MATCS was done against the traditional and the intermediate PAN + CPPS-RA approaches. Detailed evaluations of the respective case studies are presented in Chapter 5 and Chapter 6. Figure 7.1 represents the analysis of the combined evaluations of the two case studies conducted in the scope of this thesis.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Domain	**CPPS engineering**			**Software engineering**	
Approach	Traditional	(PAN) + (CPPS-RA)	MATCS	Traditional	MATCS
Knowledge representation reqirements					
R1.1 Representation of cause-effect hypotheses	--	0	++	--	++
R1.2 Representation of testing knowledge	-	-	+	-	+
R1.3 Knowledge querying and iterative extension	--	++	++	--	++
Test case specification requirements					
R2.1 Key concepts	-	0	++	-	+
R2.2 Equivalence classes concept support	--	--	++	++	++
R2.3 Risk assessment and prioritization	+	++	++	+	++
R2.4 Multi-aspect test case specification	0	0	++	-	++
R2.5 Multi-aspect test automation	--	--	0	--	0

Figure 7.1: Comparative analysis of MATCS applied at software (Company A) and CPPS (Company B) engineering environments.

Commonalities. For both groups of the domain experts the idea of deriving test cases based on the multi-disciplinary cause-effect graph was new and they both liked it very much due to the MATCS method steps reproducibility and the possibility of knowledge retrieving and extension. In both cases an efficient set of test cases was achieved thanks to the combination of FMEA, risk regions and equivalence classes. The study results indicate that MATCS approach is more efficient for analyzing quality risks once the high quality PAN has been defined and makes experimenting for system improvements more systematic compared to a traditional approach. In both cases system improvements have been revealed based on the test run results. The Company A has implemented the performance notification improvement and documented the security improvement for the future. The Company B considers implementation of the performance improvement in the scope of a bigger improvement project for the future using TDSE approach. The support of the test case automation was evaluated as partly fulfilled by both of the groups. MATCS approach offers clear description on how to define the test case automaton scope and limitations, and implement Gherkin automation scripts. However, for the domain experts it is still hard to imagine how to automate and schedule such test cases in reality due to the current implementations of their systems.

Differences. The main differences are familiarity of the domain experts with the domain specific approaches which were combined in MATCS approach and the test case specification time. The domain experts from Company A use equivalence class partitioning technique on daily basis. The domain experts from Company B have never used it before and were positively surprised how ranges of values instead of single values make testing more efficient in context in comparison to single values testing. The domain experts from Company A have never used FMEA approach, instead unstructured brainstormings were done. Therefore the Company A domain experts were very interested in application of the approach for their use case. The domain experts from the Company B conduct FMEA steps often. Risk assessment and prioritization topic was nothing new

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

for the domain experts from the Company B due to their experience with the approach from the above mentioned pilot project. The domain experts from the Company A found the MATCS risk assessment and prioritization part more structured than their traditional approach. Table of MATCS key concepts was well received by both groups, however, the domain experts from the software engineering domain have better organized legends on their engineering plans than their colleagues from Company B. Finally, since the domain experts from the Company B are familiar with the PAN concepts, it took them much less time to build the engineering network. This means that initial effort is needed to setup a high quality PAN prior test case specification phase.

7.2 Discussion of the Results

In this section all three Research Issues (RIs) stated in Section 3.1 are discussed, including the approaches to their solutions, results and evaluations.

7.2.1 RI1: Risk drivers for multi-aspect testing of system requirements.

The aim of the first RI was to investigate together with the domain experts from different disciplines identified and combined from the related literature sub-sets of risk drivers for different software-intensive engineering contexts, such as software Engineering and Cyber-physical Production Systems (CPPS) engineering. Knowing the system requirements which motivate risks is very important for quality managers, quality engineers and other domain experts, as this knowledge allows the domain experts to categorize risks [Felderer and Schieferdecker, 2014] and encapsulate the categorization into the *Multi-aspect Test Case Specification (MATCS)* approach. Furthermore, risks categorization enables better communication within and across the organizational units and allows engaging the domain experts with appropriate knowledge. RI1 motivated *RQ1: Which risk drivers motivate multi-aspect risk-based testing of cause-effect hypotheses between production process characteristics and production resource parameters/influences?*

The answer to the RQ1 is based on the discussions in scope of workshops with domain experts from different engineering disciplines and presented in *Table 4.4*. Once the most critical risk drivers have been identified, two use cases have been elicited based on the set of risk drivers: *software engineering use case Algorithm performance* and *CPPS engineering use case Aluminium surface cleaning process* (described in Chapter 5 and Chapter 6). Both groups of representatives from Company A and Company B decided that validation of hypotheses for establishment of countermeasures for timing related issues and support of timing related improvements are the best scenarios to use the multi-aspect testing approach for.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

7.2.2 RI2: Representation of multi-aspect cause-effect hypotheses.

RI2 was motivated by the problematic of representation of cause-effect hypotheses related implicit knowledge between production process characteristics and production resource parameters/influences in heterogeneous multi-disciplinary engineering environments. The RI2 motivated the *RQ2: What knowledge model can represent multi-aspect cause-effect hypotheses between production process characteristics and production resource parameters/influences?*

In order to answer RQ2, *software and CPPS engineering domain knowledge models*, Product Process Resource Asset Networks (PAN [Winkler et al., 2021]), based on the *MATCS meta-model* (Figure 4.6) were implemented in Neo4j [1] in cooperation with the domain experts from the Company A and Company B, highlighting the engineering network characteristics of the use cases (described in Chapter 5 and Chapter 6) elicited based on the outcome of the RQ1. The MATCS meta-model is built on the CPPS-RA meta-model [Biffl et al., 2021] and contains, in addition to the cause-effect, Failure Mode and Effects Analysis (FMEA) and Product Process Resource (PPR) aspects, an additional important testing layer for cause-effect hypotheses validation. The testing layer contains the cause definitions which include the equivalence class values, information about possibility and levels of asset property control and measurement, as well as levels of execution. The domain knowledge models can be queried and iteratively extended using the simple easily understandable for experts and non-experts Neo4j syntax.

The concept of PAN has been proven to be relevant and usable within the CPPS engineering, such as [Kropatschek et al., 2022], [Meixner et al., 2022], [Biffl et al., 2021], and chemical engineering [Grzymek, 2022] domains. However, based on the literature research, the first attempt to look at the software engineering domain from the CPPS point of view has been done in this thesis. The previous research in the software engineering area focused on unifying implicit knowledge purely on software and software related artefacts, such as requirements, modeling and programming languages, in a knowledge model with the ability of cause-effect paths representation Meier et al. [2019]. On the other hand, possibilities of collecting testing related knowledge across multiple disciplines has been investigated by [Chabot et al., 2016]. The method focuses on multi-physics sub-systems purely: including components with mechanical, electrical, thermal, electromagnetic physical features, but does not consider hardware/software controlling elements. MATCS approach is capable of covering both research gaps. It means that MATCS on one hand can be applied in software engineering environments and include the underlying hardware systems, while on the other hand it can support representation (and validation, see RI3 in 7.2.3) of cause-effect hypotheses in heterogeneous software-driven multi-disciplinary systems. Even though the applicability of MATCS has been investigated in contexts of two exemplary use cases representing processes, it showed promising results. This fact allows the assumption that the solution can involve more elements of the engineering network and be can applied in the full domains.

[1] Neo4j tool: `https://neo4j.com/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

7.2.3 RI3: Multi-aspect test case specification and automation to validate cause-effect relationships.

The following challenges related to validation of cause-effect hypotheses in multi-disciplinary engineering environments motivated RI3: in addition to the lack of sufficient multi-disciplinary knowledge representation, it is insufficient business and code coverage, as well as resource availability and time limitations. The knowledge representation part has been resolved by design and implementation of the knowledge model described in RI2 in 7.2.2. The other challenges motivated the *RQ3: What are the process steps to support multi-aspect testing in CPPS and Software engineering?*

In order to validate the cause-effect hypotheses and address the testing challenges *MATCS method* which allows to specify the relevant minimal set of test cases with the minimum amount of test data has been designed (Chapter 4), applied in the two case studies and evaluated (described in Chapter 5 and Chapter 6). The method as well supports the quality experts with scope and limitations for the test case automation. MATCS method is conducted in three steps: 1) domain relevant PAN based on [Winkler et al., 2021] (RQ2); 2) CPPS Risk Assessment (CPPS-RA) which allows to define the cause-effect hypotheses based on the risks and to prepare the cause-effect graph (extended PAN based on [Biffl et al., 2021]), extended with risk drivers definition step (RQ1) and risk regions identification step to limit the amount of specified test cases for validation of the hypotheses; 3) MATCS which explains how to derive test cases from the cause-effect graph designed in step 2 and define the minimum amount of test data using the equivalence classes partitioning approach [Burnstein, 2003], as well as how to set the scope and limitations for test case automation and gives an answer to *RQ3a*. MATCS method together with the risk drivers identification step and MATCS meta-model addresses the cause-effect hypotheses validation challenges in multi-disciplinary environments, such as implicit distributed system knowledge [Meier et al., 2020, Biffl et al., 2021], as well as resource availability and time limitations [Felderer and Ramler, 2014]. Test strategy discussed in [Li and Kang, 2015] provides support for testing and evaluation of CPS, however, is designed for reliability testing only and is limited to hardware/software components and the network architecture. The needed support discussed in [Meixner et al., 2020] for test case automation in multi-disciplinary engineering environments has been provided by the method, which answers *RQ3b*.

7.3 Limitations

This section is designed in order to discuss the limitations of this research that were not possible to overcome in the scope of this work, with regards to the following aspects: data collection, use case complexity, applicability of the method, and solution scope.

Data collection. Taking into consideration the fact that selection of the partner companies was based on the context of the research project and availability, it could be assumed that within the study availability and selection bias could exist. Furthermore, the number of domain experts participating in each case study representing different

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

roles was rather limited. Some of the domain experts represented different roles, which could lead to scarce representation of the engineering roles.

Limited complexity of the selected for evaluation use cases. Both of the selected use cases represent real World scenarios. The complexity of the scenarios is sufficient for evaluation of the novel method and real improvements of the systems have been achieved. The evaluation has been conducted by a limited amount of domain experts who often represented multiple engineering roles. Other domain experts might have had other results which could lead have lead to different conclusions.

Limited applicability of the method. Based on the discussion with the domain experts and definition of risk drivers, two use cases were elicited. The method has been validated and evaluated for such scenarios as defect detection and system improvements during production phase. All scenarios were motivated by performance risk driver. More risk drivers could be investigated in relation to the multi-aspect testing, which could affect the output of the thesis. For example, security aspect could present certain interest, as well as application of the method for initial test driven system design.

Limited scope of the solution. The method presented in this work is based on the requirements from two domains: software engineering and CPPS engineering. The assumption behind the topic of the thesis is that the quality engineer needs to receive data from other multiple domain experts, such as software, hardware, mechanical and electrical engineers, in order to assure the quality on their level. The method therefore could also be evaluated in other multi-disciplinary contexts, such as, for instance, aerospace or chemical engineering.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 8

Conclusion and Future Work

In this chapter the main findings of this work are concluded. Furthermore, a summary of the contributions to research is presented. Last, an overview for potential future work is given.

8.1 Conclusion

This thesis was motivated by the problematic of representation and validation of cause-effect hypotheses between production process characteristics and production resource parameters/influences in heterogeneous multi-disciplinary engineering environments, including the set of standard testing challenges (see Chapter 1). Constantly growing complexity of modern software-intensive production systems and their heterogeneous nature makes it challenging for individual experts with different engineering backgrounds to understand the details of the system and their processes. The quality engineers are not an exception, since they need to have knowledge about aspects of the system from multiple disciplines, such as product, process and resource (PPR) specifications, as well as environmental influences, in order to specify test cases for cause-effect hypotheses validation. Such knowledge is often implicit and spread across the domain experts, which adds even more complexity to the already existing testing challenges, such as inefficient coverage of the system parts and their characteristics, as well as resource availability and time limitations. Automation of such test cases is often not possible or limited due to heterogeneous tools, artifacts, and systems. Based on the literature survey and workshops with domain experts from cyber-physical production systems (CPPS) and software engineering domains, requirements for the novel MATCS approach have been elicited. Following, the MATCS approach has been designed and evaluated with regards to the TDSE approach in two case studies with the above mentioned domain experts. The results of the work regarding the research questions are summarized in the following text.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The answer to the *(RQ1) Which risk drivers motivate multi-aspect risk-based testing of cause-effect hypotheses between production process characteristics and production resource parameters/influences?* is a sub-set of system requirements from [ISO25030, 2019] limited by the system requirements from the related literature and reduced one more time after the evaluation of the initial sub-set by the domain experts who later participated in the two case studies. Based on the final set of critical risk drivers, the two use cases for the case studies have been elicited.

To answer *(RQ2) What knowledge model can represent multi-aspect cause- effect hypotheses between production process characteristics and production resource parameters/influences?* Based on PAN [Winkler et al., 2021], CPPS-RA [Biffl et al., 2021] and additional elicited from literature and proposed by domain experts requirements, MATCS meta-model has been designed. The meta-model contains the Failure Modes and Effects Analysis (FMEA) aspect to support systematic and efficient risk management, PPR Asset Network (PAN) aspect to describe a CPPS structure, cause-effect analysis aspect to describe the relationships between effects and their causes and finally testing aspect to extend the domain models with test case knowledge. Two domain knowledge models, such as CPPS and software engineering knowledge models, have been derived from the meta-model and implemented. The domain knowledge models allow the domain experts to make their implicit knowledge explicit and propagate the cause-effect relationships through the PAN. The resulting models cover the scientific gaps identified from the literature, such as taking into consideration only software engineering knowledge [Meier et al., 2019] and investigating only multi-physics sub-systems, including mechanical, electrical, thermal and electromagnetic features [Chabot et al., 2016].

For the *(RQ3) What are the process steps to support multi-aspect testing in CPPS and software engineering?* The five-steps MATCS method has been introduced based on elicited from literature and proposed by domain experts requirements. The resulting method covers the research gap identified from [Li and Kang, 2015], such as allowing to cover other than reliability testing only. *RQ3a: What is the minimum set of test cases and test data a test engineer requires to validate a hypothesis based on cause-effect relationships?* was answered in the following way: based on the MATCS domain knowledge models, FMEA approach, the hypotheses from the CPPS-RA [Biffl et al., 2021], and Equivalence Partitioning Testing (EPT) which divides the input domain into classes of data, in order to validate cause-effect hypotheses displayed as cause-effect relationships in the PAN, MATCS method allows the domain experts to derive the minimum set of test cases needed for the hypotheses validation without lowering the results quality. For the *RQ3b: What are the test scope and limitations for multi-aspect test automation?* Step 3.3 (Section 4.5.3) of MATCS method was designed to define the scope and limitations for automation of the sub-set of the test cases.

All three artefacts together create one MATCS approach which was evaluated against the elicited requirements in CPPS and software engineering domain case studies. Following, a comparative analysis of the combined results has been conducted. The study results indicate that initial effort is needed to setup a high quality PAN, however reusing the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

PAN will reduce the multi-aspect test case specification, automation, data collection and analysis effort in the future. The initial test space in multi-disciplinary engineering domains is usually very complex and therefore needs to be reduced efficiently due to limited resources for testing. MATCS allows saving testing resources without lowering the quality of the results. MATCS approach is more efficient for analyzing quality risks once a high quality PAN has been defined and makes experimenting for system improvements more systematic compared to a traditional approach. Furthermore, MATCS approach can be used for documentation of system requirements in form of test cases and therefore can support both, engineering first approach and TDSE based on the software TDD approach.

8.2 Future Work

The novel method could be applied under a range of different conditions and leaves numerous possibilities for extension. In this section possible future work is suggested, based on the discussion and limitations from the previous chapters and ideas which were not in the scope of this thesis.

Method applicability. It should be investigated whether the novel method can be applied for use cases elicited based on different than presented in this work risk drivers.

Scope of the solution. While the novel approach is applicable for the elicited use cases in software and CPPS engineering domains, it should be clarified whether the method can be applicable for the other use cases in the same and other engineering domains.

Critical risk regions measurement. It is worth investigating how preconditions for test cases which contain critical values that cannot be executed due to the possible system or product damage, can be measured to avoid risks.

Improvement test cases which executions are limited by hardware. Some of the proposed improvements cannot be executed due to the limitations in hardware specifications. It should be investigated, whether simulations would be capable of execution of such test cases and whether the test results would be reliable.

Automated test case generation. Automated generation of the test cases which parameters are stored in the domain knowledge models should be investigated.

Versioning of the knowledge models. The failure mode from the FMEA usually has dependency on a failure from PAN. For each FMEA a snapshot of the PAN is used. The values from the snapshot are stored back to the knowledge model. It could potentially happen that the same parameter values from PAN, in combination with the other parameters, could lead to different effects. In MATCS approach the versioning of FMEA is not considered.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Company A - Software domain knowledge model code

Company A experiment - data

CPU	Run 1 (ms)	Run 2 (ms)	Run 3 (ms)	Run 4 (ms)
CPU 3-6%	0.094	0.091	0.09	0.088
CPU 78-81%	0.106	0.092	0.144	0.094
CPU 100%	0.353	0.311	0.363	0.416

Figure 1: Company A - Data: algorithm performance time behaviour depending on the CPU load.

Company A experiment - knowledge model code.

In order to generate the Company A knowledge model, copy the code below and execute it in Neo4j[1] software.

```
// Product:PPRAssets
MERGE (`SetupCode`:Product:PPRAsset {`name`:"Setup code"})
MERGE (`TestCode`:Product:PPRAsset {`name`:"Test code"})
MERGE (`TestRunResult`:Product:PPRAsset {`name`:"Test run
    result"})
MERGE (`TestOutput`:Attribute:PPRAttribute {`name`:"Output"})
MERGE (`TestRunResult`)-[:has_Attribute]-(`TestOutput`)
MERGE (`OutputTimeout`:OutputRiskyTimeout {`name`:"Risky",`
    value`:"353ms"})
MERGE (`OutputOK`:OutputOK {`name`:"OK",`value`:"[1-200ms]"})
MERGE (`TestOutput`)-[:has_Value]-(`OutputTimeout`)
MERGE (`TestOutput`)-[:has_Value]-(`OutputOK`)

// Process:PPRAssets
MERGE (`VMrun`:Process:PPRAsset {`name`:"Execute test case on a
     virtual machine"})
```

[1]Neo4j tool: https://neo4j.com/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
MERGE (`RunTestCase`:Process:PPRAsset {`name`:"Run test case
    code"})

// Product-Process:PPRAssets Links
MERGE (`VMrun`)-[:`has_Input`]->(`SetupCode`)
MERGE (`RunTestCase`)-[:`has_Input`]->(`TestCode`)
MERGE (`RunTestCase`)-[:`has_Input`]->(`VMrun`)
MERGE (`RunTestCase`)-[:`has_Ouput`]->(`TestRunResult`)

// Test Resource:PPRAssets
MERGE (`VMtest`:Resource:PPRAsset {`name`:"Virtual machine test
    "})
MERGE (`RAMtest`:Resource:PPRAsset {`name`:"RAM test"})
MERGE (`CPUtest`:Resource:PPRAsset {`name`:"CPU test"})
MERGE (`OStest`:Resource:PPRAsset {`name`:"OS test"})
MERGE (`HypervisorTest`:Resource:PPRAsset {`name`:"Hypervisor
    test"})
MERGE (`HostTest`:Resource:PPRAsset {`name`:"Host machine test
    "})
MERGE (`RamHostTest`:Resource:PPRAsset {`name`:"RAM host
    machine test"})
MERGE (`CPUhostTest`:Resource:PPRAsset {`name`:"CPU host
    machine test"})

// Test Process-Resource:PPRAssets Links
MERGE (`VMrun`)-[:`has_Resource`]-(`VMtest`)

// Test has_Software:PPRAssets Links
MERGE (`VMtest`)-[:`has_Software`]-(`RAMtest`)
MERGE (`VMtest`)-[:`has_Software`]-(`CPUtest`)
MERGE (`VMtest`)-[:`has_Software`]-(`OStest`)
MERGE (`CPUtest`)-[:`has_Software`]-(`HypervisorTest`)

// Test has_Hardware:PPRAssets Links
MERGE (`HypervisorTest`)-[:`has_Hardware`]-(`HostTest`)
MERGE (`HostTest`)-[:`has_Hardware`]-(`RamHostTest`)
MERGE (`HostTest`)-[:`has_Hardware`]-(`CPUhostTest`)

// Service Resource:PPRAssets
MERGE (`WebServiceAPI`:Resource:PPRAsset {`name`:"Web service
    API"})
MERGE (`CodeVersion`:Attribute:PPRAttribute {`name`:"Version",`
    Version`:"10.1.24"})
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
MERGE (`WebServiceAPI`) -[:has_Attribute]-(`CodeVersion`)
MERGE (`VMservice`:Resource:PPRAsset {`name`:"Virtial machine
    service"})
MERGE (`RAMservice`:Resource:PPRAsset {`name`:"RAM service"})
MERGE (`CPUservice`:Resource:PPRAsset {`name`:"CPU service"})
MERGE (`CPULoad`:Attribute:PPRAttribute {`name`:"Load", `value
    `:"[88%]"})
MERGE (`CPUservice`) -[:has_Attribute]-(`CPULoad`)
MERGE (`OSservice`:Resource:PPRAsset {`name`:"OS service"})
MERGE (`HypervisorService`:Resource:PPRAsset {`name`:"
    Hypervisor service"})
MERGE (`CoreAmount`:Attribute:PPRAttribute {`name`:"Cores", `
    value`:">=2"})
MERGE (`HypervisorService`) -[:has_Attribute]-(`CoreAmount`)
MERGE (`HostService`:Resource:PPRAsset {`name`:"Host machine
    service"})
MERGE (`RamHostService`:Resource:PPRAsset {`name`:"RAM host
    machine service"})
MERGE (`CPUhostService`:Resource:PPRAsset {`name`:"CPU host
    machine service"})

// Service Process-Resource:PPRAssets Links
MERGE (`RunTestCase`) -[:`has_Resource`]-(`WebServiceAPI`)

// Service has_Software:PPRAssets Links
MERGE (`WebServiceAPI`) -[:`has_Software`]-(`VMservice`)
MERGE (`VMservice`) -[:`has_Software`]-(`RAMservice`)
MERGE (`VMservice`) -[:`has_Software`]-(`CPUservice`)
MERGE (`VMservice`) -[:`has_Software`]-(`OSservice`)
MERGE (`CPUservice`) -[:`has_Software`]-(`HypervisorService`)

// Service has_Hardware:PPRAssets Links
MERGE (`HypervisorService`) -[:`has_Hardware`]-(`HostService`)
MERGE (`HostService`) -[:`has_Hardware`]-(`RamHostService`)
MERGE (`HostService`) -[:`has_Hardware`]-(`CPUhostService`)

// Service has_PPRDependency:PPRAssets Links
MERGE (`TestOutput`) -[:`has_PPRDependency`]->(`CPULoad`)

// Cause:FMEAAsset
MERGE (`CoresSettingWrong`:Cause:FMEAAsset {`name`:"Cores
    setting wrong"})
MERGE (`Cores`:Attribute:FMEAAttribute {`name`:"Cores",`
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
    CanMeasure‘:"No",‘CanControl‘:"No",‘LevelsOfControl‘:"Domain
     expert",‘LevelOfExecution‘:"Test"})
MERGE (‘CoresOK‘:EquivalenceClassOK {‘name‘:"OK",‘value
    ‘:">=2"})
MERGE (‘CoresFew‘:EquivalenceClassFew {‘name‘:"Low",‘value
    ‘:"<2"})
MERGE (‘CoresSettingWrong‘)-[:has_Attribute]-(‘Cores‘)
MERGE (‘Cores‘)-[:has_EQClass]-(‘CoresOK‘)
MERGE (‘Cores‘)-[:has_EQClass]-(‘CoresFew‘)

MERGE (‘NotEnoughCPU‘:Cause:FMEAAsset {‘name‘:"Not enough CPU
    "})
MERGE (‘Load‘:Attribute:FMEAAttribute {‘name‘:"Load",‘
    CanMeasure‘:"Yes, auto",‘CanControl‘:"Yes, auto",‘
    LevelsOfControl‘:"Domain expert",‘LevelOfExecution‘:"Test"})
MERGE (‘LoadOK‘:EquivalenceClassOK {‘name‘:"OK",‘value
    ‘:"[3-80%]"})
MERGE (‘LoadHigh‘:EquivalenceClassHigh {‘name‘:"High",‘value
    ‘:"[81-100%]"})
MERGE (‘NotEnoughCPU‘)-[:has_Attribute]-(‘Load‘)
MERGE (‘Load‘)-[:has_EQClass]-(‘LoadOK‘)
MERGE (‘Load‘)-[:has_EQClass]-(‘LoadHigh‘)

MERGE (‘NOKcodeVersion‘:Cause:FMEAAsset {‘name‘:"NOK code
    version"})
MERGE (‘Version‘:Attribute:FMEAAttribute {‘name‘:"Version"})
MERGE (‘NOKcodeVersion‘)-[:has_Attribute]-(‘Version‘)
MERGE (‘VersionOK‘:EquivalenceClassOK {‘name‘:"OK",‘value
    ‘:"[1.0.0-10.1.22,10.1.24]"})
MERGE (‘VersionNOK‘:EquivalenceClassNOK {‘name‘:"NOK",‘value
    ‘:"[10.1.23,?]"})
MERGE (‘Version‘)-[:has_EQClass]-(‘VersionOK‘)
MERGE (‘Version‘)-[:has_EQClass]-(‘VersionNOK‘)

// Fault:FMEAAsset
MERGE (‘Timeout‘:Fault:FMEAAsset {‘name‘:"Data is not delivered
     on time"})
MERGE (‘Output‘:Attribute:FMEAAsset {‘name‘:"Output"})
MERGE (‘Timeout‘)-[:has_Attribute]-(‘Output‘)
MERGE (‘ECOutputOK‘:EquivalenceClassOK {‘name‘:"OK",‘value
    ‘:"[1-200ms]"})
MERGE (‘ECTimeout‘:EquivalenceRiskyTimeout {‘name‘:"Risky",‘
    value‘:"[201-500ms]"})
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
MERGE (`Output`)-[:has_EQClass]-(`OutputOK`)
MERGE (`Output`)-[:has_EQClass]-(`ECTimeout`)

// Cause-Effect:FMEAAsset Links
MERGE (`CoresSettingWrong`)-[:`Cause-Effect`]-(`NotEnoughCPU`)
MERGE (`NotEnoughCPU`)-[:`Cause-Effect`]-(`NOKcodeVersion`)
MERGE (`NOKcodeVersion`)-[:`Cause-Effect`]->(`Timeout`)

// has_FMEADependency:FMEAAsset Links
MERGE (`Load`)-[:`has_FMEADependency`]-(`Cores`)
MERGE (`Load`)-[:`has_FMEADependency`]-(`CPULoad`)
MERGE (`Version`)-[:`has_FMEADependency`]-(`CodeVersion`)
MERGE (`Cores`)-[:`has_FMEADependency`]-(`CoreAmount`)
MERGE (`OutputTimeout`)-[:`has_FMEADependency`]-(`ECTimeout`)
MERGE (`OutputOK`)-[:`has_FMEADependency`]-(`ECOutputOK`)
MERGE (`ECTimeout`)-[:`has_FMEADependency`]-(`LoadHigh`)
MERGE (`ECOutputOK`)-[:`has_FMEADependency`]-(`LoadOK`)
```

In order to display the knowledge graph, execute:

```
MATCH (n)
RETURN n
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Company B - CPPS domain knowledge model code

Company B experiment - data

Run	Temperature of water in tank 1 (C°)	Washing time (sec)
1	67	202
2	68	201
3	68	200
4	68	201
5	69	202
6	68	202
7	68	202
8	69	201
9	69	200
10	68	202
11	67	200
12	68	202
13	69	200
14	67	200
15	69	201
16	69	202
17	67	201
18	68	201
19	68	202
20	69	200

Figure 2: Company B - Data: aluminium surface cleaning performance time behaviour depending on the water temperature.

Company B experiment - knowledge model code.

In order to generate the Company B knowledge model, copy the code below and execute it in Neo4j[2] software.

[2]Neo4j tool: `https://neo4j.com/`

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
// Product:PPRAssets
MERGE (`DistilledWater`:Product:PPRAsset {`name`:"Distlled
    water"})
MERGE (`AluminiumPart`:Product:PPRAsset {`name`:"Aluminium part
    "})
MERGE (`ChemistryAandB`:Product:PPRAsset {`name`:"Chemistry A
    and B"})
MERGE (`AluminiumPartOutput`:Product:PPRAsset {`name`:"
    Aluminium part degreased"})
MERGE (`Grease`:Attribute:PPRAttribute {`name`:"Grease"})
MERGE (`AluminiumPartOutput`)-[:has_Attribute]-(`Grease`)

// Process:PPRAssets
MERGE (`Degreasing`:Process:PPRAsset {`name`:"Degreasing"})
MERGE (`DegreasingTime`:Attribute:PPRAttribute {`name`:"
    Degreasing time"})
MERGE (`Degreasing`)-[:has_Attribute]-(`DegreasingTime`)
MERGE (`DegreasingTimeDecreased`:DegreasingTimeImproved {`name
    `:"Degreasing time decreased",`value`:"[200-202s]"})
MERGE (`DegreasingTime`)-[:has_Value]-(`DegreasingTimeDecreased
    `)

// Product-Process:PPRAssets Links
MERGE (`Degreasing`)-[:`has_Input`]->(`DistilledWater`)
MERGE (`Degreasing`)-[:`has_Input`]->(`AluminiumPart`)
MERGE (`Degreasing`)-[:`has_Input`]->(`ChemistryAandB`)
MERGE (`Degreasing`)-[:`has_Ouput`]->(`AluminiumPartOutput`)

// Resource:PPRAssets
MERGE (`BagFilter`:Resource:PPRAsset {`name`:"Bag filter"})

MERGE (`ChipFilter`:Resource:PPRAsset {`name`:"Chip filter"})
MERGE (`Roller`:Resource:PPRAsset {`name`:"Roller"})

MERGE (`Chamber`:Resource:PPRAsset {`name`:"Chamber"})
MERGE (`Nozzle`:Resource:PPRAsset {`name`:"Nozzle"})
MERGE (`PressureMeasurmentSensor`:Resource:PPRAsset {`name`:"
    Pressure measurment sensor"})
MERGE (`TemperatureMeasurmentSensor`:Resource:PPRAsset {`name
    `:"Temperature measurment sensor"})
MERGE (`Pump`:Resource:PPRAsset {`name`:"Pump"})
MERGE (`LevelMeasurmentSensor`:Resource:PPRAsset {`name`:"Level
     measurment sensor"})
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
MERGE (`WashTank`:Resource:PPRAsset {`name`:"Wash tank"})
MERGE (`AluminiumConcentration`:Attribute:PPRAttribute {`name
    `:"Aluminium concentration"})
MERGE (`WashTank`)-[:has_Attribute]-(`AluminiumConcentration`)
MERGE (`Freshness`:Attribute:PPRAttribute {`name`:"Freshness"})
MERGE (`WashTank`)-[:has_Attribute]-(`Freshness`)
MERGE (`phLevel`:Attribute:PPRAttribute {`name`:"pH level"})
MERGE (`WashTank`)-[:has_Attribute]-(`phLevel`)

MERGE (`OilSeparator`:Resource:PPRAsset {`name`:"Oil separator
    "})

MERGE (`HeatingElement`:Resource:PPRAsset {`name`:"Heating
    element"})
MERGE (`HeatingTemperature`:Attribute:PPRAttribute {`name`:"
    Heating temperature"})
MERGE (`HeatingElement`)-[:has_Attribute]-(`HeatingTemperature
    `)
MERGE (`HeatingTemperatureHigh`:HeatingTemperature {`name`:"
    Heating temperature high",`value`:"[66-70C]"})
MERGE (`HeatingTemperatureLow`:HeatingTemperature {`name`:"
    Heating temperature low",`value`:"[60-65C]"})
MERGE (`HeatingTemperature`)-[:has_EQClass]-(`
    HeatingTemperatureLow`)
MERGE (`HeatingTemperature`)-[:has_EQClass]-(`
    HeatingTemperatureHigh`)

MERGE (`ConductivityMeasurmentDevice`:Resource:PPRAsset {`name
    `:"Conductivity measurment device"})
MERGE (`ConcentrationMeasurmentDevice`:Resource:PPRAsset {`name
    `:"Concentration measurment device"})

MERGE (`TransportSystem`:Resource:PPRAsset {`name`:"Transport
    system"})

// Resource:ITAsset
MERGE (`OPCUA`:Resource:ITAsset {`name`:"OPC UA"})

// Process-Resource:PPRAssets Links
MERGE (`Degreasing`)-[:`has_Resource`]-(`BagFilter`)
MERGE (`Degreasing`)-[:`has_Resource`]-(`ChipFilter`)
MERGE (`Degreasing`)-[:`has_Resource`]-(`Chamber`)
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
MERGE (`Degreasing`) -[:`has_Resource`]-(`WashTank`)
MERGE (`Degreasing`) -[:`has_Resource`]-(`TransportSystem`)

// has_Mechanical:PPRAssets Links
MERGE (`ChipFilter`) -[:`has_Mechanical`]-(`Roller`)
MERGE (`Chamber`) -[:`has_Mechanical`]-(`Nozzle`)
MERGE (`Chamber`) -[:`has_Mechanical`]-(`Pump`)
MERGE (`WashTank`) -[:`has_Mechanical`]-(`OilSeparator`)
MERGE (`WashTank`) -[:`has_Mechanical`]-(`HeatingElement`)

// has_Electrical:PPRAssets Links
MERGE (`Chamber`) -[:`has_Electrical`]-(`LevelMeasurmentSensor`)
MERGE (`WashTank`) -[:`has_Electrical`]-(`
    ConductivityMeasurmentDevice`)
MERGE (`WashTank`) -[:`has_Electrical`]-(`
    ConcentrationMeasurmentDevice`)
MERGE (`Nozzle`) -[:`has_Electrical`]-(`PressureMeasurmentSensor
    `)
MERGE (`Nozzle`) -[:`has_Electrical`]-(`
    TemperatureMeasurmentSensor`)

// Controller-Resource:PPRAssets Links
MERGE (`HeatingElement`) -[:`has_Controller`]-(`OPCUA`)

// has_PPRDependency:PPRAssets Links
MERGE (`DegreasingTimeDecreased`) -[:`has_PPRDependency`]->(`
    HeatingTemperatureHigh`)

// Cause:FMEAAsset
MERGE (`TemperatureIncreased`:Cause:FMEAAsset {`name`:"
    Temperature increased"})
MERGE (`Temperature`:Attribute:FMEAAttribute {`name`:"
    Temperature",`CanMeasure`:"Yes",`CanControl`:"Yes",`
    LevelsOfControl`:"Domain expert",`LevelOfExecution`:"Prod"})
MERGE (`TemperatureOK`:EquivalenceClassOK {`name`:"Washing time
     before",`value`:"[60-66C]"})
MERGE (`TemperatureImpr`:EquivalenceClassImpr {`name`:"Washing
    time improved",`value`:"[67-69C]"})
MERGE (`TemperatureCr`:EquivalenceClassCr {`name`:"Critical",`
    value`:"[70-71C]"})
MERGE (`TemperatureIncreased`) -[:has_Attribute]-(`Temperature`)
MERGE (`Temperature`) -[:has_EQClass]-(`TemperatureOK`)
MERGE (`Temperature`) -[:has_EQClass]-(`TemperatureImpr`)
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
MERGE (`Temperature`)-[:has_EQClass]-(`TemperatureCr`)

MERGE (`TankContentFresh`:Cause:FMEAAsset {`name`:"Tank content
    fresh"})
MERGE (`DetergentConcentrationIncreased`:Cause:FMEAAsset {`name
    `:"Detergent concentration increased"})
MERGE (`AluminiumConcentrationDecreased`:Cause:FMEAAsset {`name
    `:"Aluminium concentration decreased"})
MERGE (`AmountOfGreaseLow`:Cause:FMEAAsset {`name`:"Amount of
    grease low"})

// Improvement:FMEAAsset
MERGE (`TimeDecreased`:Cause:FMEAAsset {`name`:"Aluminium
    surface cleaning cycle time decreased"})
MERGE (`Time`:Attribute:FMEAAsset {`name`:"Time"})
MERGE (`TimeDecreased`)-[:has_Attribute]-(`Time`)
MERGE (`ECLowTemp`:EquivalenceLowTemp {`name`:"Avg low
    temperature",`value`:"225s"})
MERGE (`ECHighTemp`:EquivalenceHighTemp {`name`:"Avg high
    temperature",`value`:"201s"})
MERGE (`ECCritTemp`:EquivalenceCritTemp {`name`:"Critical
    temperature"})
MERGE (`Time`)-[:has_EQClass]-(`ECLowTemp`)
MERGE (`Time`)-[:has_EQClass]-(`ECHighTemp`)
MERGE (`Time`)-[:has_EQClass]-(`ECCritTemp`)

// Cause-Effect:FMEAAsset Links
MERGE (`TemperatureIncreased`)-[:`Cause-Effect`]-(`
    TimeDecreased`)
MERGE (`AmountOfGreaseLow`)-[:`Cause-Effect`]-(`TimeDecreased`)
MERGE (`TankContentFresh`)-[:`Cause-Effect`]-(`
    DetergentConcentrationIncreased`)
MERGE (`DetergentConcentrationIncreased`)-[:`Cause-Effect`]-(`
    TimeDecreased`)
MERGE (`AluminiumConcentrationDecreased`)-[:`Cause-Effect`]-(`
    TimeDecreased`)

// has_FMEADependency:FMEAAsset Links
MERGE (`AmountOfGreaseLow`)-[:`has_FMEADependency`]-(`Grease`)
MERGE (`TemperatureIncreased`)-[:`has_FMEADependency`]-(`
    HeatingTemperature`)
MERGE (`TankContentFresh`)-[:`has_FMEADependency`]-(`phLevel`)
MERGE (`DetergentConcentrationIncreased`)-[:`has_FMEADependency
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
    `]-(`Freshness`)
MERGE (`AluminiumConcentrationDecreased`)-[:`has_FMEADependency
    `]-(`AluminiumConcentration`)
```

In order to display the knowledge graph, execute:

```
MATCH (n)
RETURN n
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acronyms

API Application Programming Interface. 72–76, 78

CEN CPPS Engineering Network. 21–23, 92–94, 115, 119

CPPS Cyber-Physical Production System. xiv, 6–10, 12, 14, 15, 17–19, 21, 22, 30, 33, 38–40, 42–45, 48, 50, 51, 82, 85, 92, 93, 101, 102, 105–110, 112, 115–117

CPPS-RA CPPS Risk Assessment. 21–24, 32, 46, 47, 49, 53, 54, 63, 71, 83, 89, 91, 92, 100–103, 105, 108, 109, 112, 115, 120

CPS Cyber-Physical System. 15

CPU Central Processing Unit. 3, 73–78, 81, 116

CS Case Study. 37, 38

CTU Czech Technical University. 16

DSR Design Science Research. 34

EC Equivalence Class. 75, 95, 119, 120

ECP Equivalence Class Partitioning. 27, 43

EN Engineering Network. 48, 49, 52, 53, 57, 60–62, 116

ETFA Emerging Technologies and Factory Automation. 8, 40, 60

FMEA Failure Modes and Effects Analysis. 19–23, 43, 45, 49, 51, 53, 60, 62, 63, 66, 72, 74, 82, 83, 91, 93, 97, 99, 102, 106, 108, 115, 116, 119

I4.0 Industry 4.0. 16

IDEF0 Integration Definition for Process Modelling. 22, 23, 47, 49, 50, 53, 55, 115, 116

IEC International Electrotechnical Commission. 10, 11, 50, 116, 119

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

IEEE Institute of Electrical and Electronics Engineers. 8, 40, 60

IIoT Industrial Internet of Things. 11, 12

ISO International Organization for Standardization. 10, 11, 50, 116, 119

MATCS Multi-Aspect Test Case Specification. 43–45, 47, 48, 50, 53–55, 60, 69, 74, 82–84, 89, 91, 94, 101–103, 105–109, 111–113, 115–117, 120

MDE Multi-Disciplinary Engineering. 9, 10, 18, 38

MIG Metal Inert Gas. 85

PAN PPR Asset Network. 15–17, 39, 44, 46–49, 60, 62, 63, 67, 70, 71, 74, 85, 91, 92, 100–103, 105–109, 112, 113, 115, 116, 120

pH Potential of Hydrogen. 88, 95

PPR Product, Process, Resource. 1, 10, 15, 16, 21, 44, 45, 60, 62, 63, 71, 83, 91, 92, 99, 108, 115, 116

QA Quality Assurance. 48

RA Risk Assessment. 9, 21

RBT Risk-Based Testing. 26

RI Research Issue. 31–33, 105, 107–109

RQ Research Question. 31–34, 44, 107–109, 112

SEN Software Engineering Network. 71, 73, 74, 99, 116

SLR Systematic Literature Review. 36, 37

SQuaRE Systems and software Quality Requirements and Evaluation. 10, 11, 32, 50, 116, 119

SuI System under Inspection. 21, 52, 61, 62, 72, 73, 93, 94

SuT System under Test. 5

TDD Test-driven Development. 42, 43, 113

TDSE Test-driven System Engineering. 35, 42, 81, 99, 106, 111, 113

VM Virtual Machine. 72–76

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Bibliography

Iso/iec/ieee systems and software engineering – architecture description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pages 1–46, 2011. doi: 10.1109/IEEESTD.2011.6129467.

B. S. Ahmed. Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. *Engineering Science and Technology, an International Journal*, 19(2):737–753, 2016. ISSN 2215-0986. doi: https://doi.org/10.1016/j.jestch.2015.11.006.

L. Antão, R. Pinto, J. Reis, and G. Gonçalves. Requirements for testing and validating the industrial internet of things. In *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 110–115, 2018. doi: 10.1109/ICSTW.2018.00036.

P. Baxter and S. Jack. Qualitative case study methodology: Study design and implementation for novice researchers. *Qualitative Report*, 13, 01 2010. doi: 10.46743/2160-3715/2008.1573.

K. Beck. *Test Driven Development. By Example (Addison-Wesley Signature)*. Addison-Wesley Longman, Amsterdam, 2002. ISBN 0321146530.

A. Bertolino. Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering (FOSE '07)*, pages 85–103, 2007. doi: 10.1109/FOSE.2007.25.

S. Biffl, D. Gerhard, and A. Lüder. *Introduction to the Multi-Disciplinary Engineering for Cyber-Physical Production Systems*, pages 1–24. Springer International Publishing, Cham, 2017a. ISBN 978-3-319-56345-9. doi: 10.1007/978-3-319-56345-9_1. URL `https://doi.org/10.1007/978-3-319-56345-9_1`.

S. Biffl, A. Lüder, and D. Gerhard, editors. *Multi-Disciplinary Engineering for Cyber-Physical Production Systems, Data Models and Software Solutions for Handling Complex Engineering Projects*. Springer, 2017b. ISBN 978-3-319-56344-2.

S. Biffl, M. Eckhart, A. Luder, and E. Weippl. *Security and Quality in Cyber-Physical Systems Engineering: With Forewords by Robert M. Lee and Tom Gilb*. Springer Publishing Company, Incorporated, 1st edition, 2019. ISBN 3030253112.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

S. Biffl, A. Lüder, E. Kiesling, K. Meixner, F. Rinker, C. Engelbrecht, M. Eckhart, and D. Winkler. Multi-Aspect Risk Exploration in Models for Positioning and Joining Simulation (Case Study) Part II. Technical Report CDL-SQI-2020-07, CDL-SQI, Institute for ISE, TU Wien, Nov. 2020a. https://url.tuwien.at/ujyge.

S. Biffl, A. Lueder, K. Meixner, F. Rinker, M. Eckhart, and D. Winkler. Multi-viewmodel risk assessment for positioning and joining simulation (case study). technical report cdl-sqi 2020-05. 2020b. ttps://qse.ifs.tuwien.ac.at/cdl-sqi-2020-05/.

S. Biffl, A. Lueder, K. Meixner, F. Rinker, M. Eckhart, and D. Winkler. Multi-View-Model Risk Assessment in Cyber-Physical Production Systems Engineering. In S. Hammoudi, L. F. Pires, E. Seidewitz, and R. Soley, editors, *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2021), Online, February 6-8, 2019*, pages 163–170. SciTePress, Feb. 2021. doi: 10.5220/0010224801630170.

J. Biolchini, P. Mian, A. Candida, and C. Natali. Systematic review in software engineering. 01 2005.

S. Bisht. *Robot framework test automation*. Packt Publishing Ltd, 2013.

I. Burnstein. *Practical Software Testing*. Springer Verlag, 2003. ISBN ISBN0-387-95131-8.

T. Carbone and D. Tippett. Project risk management using the project risk fmea. *Engineering Management Journal*, 16, 04 2015. doi: 10.1080/10429247.2004.11415263.

M. Chabot, L. Pierre, and A. Nabais-Moreno. A requirement driven testing method for multi-disciplinary system design. MODELS '16, page 396–405, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450343213. doi: 10.1145/2976767.2976795. URL https://doi.org/10.1145/2976767.2976795.

P. Duvall, S. M. Matyas, and A. Glover. *Continuous Integration: Improving Software Quality and Reducing Risk*. Addison-Wesley Signature Series. Addison-Wesley, Upper Saddle River, NJ, 2007. ISBN 978-0-321-33638-5. URL http://my.safaribooksonline.com/9780321336385.

H. Elmaraghy. *Changing and Evolving Products and Systems – Models and Enablers*, pages 25–45. 01 2009. ISBN 978-1-84882-066-1. doi: 10.1007/978-1-84882-067-8_2.

E. Engström, M.-A. Storey, P. Runeson, M. Höst, and M. T. Baldassarre. How software engineering research aligns with design science: a review. *Empirical Software Engineering*, 25(4):2630–2660, 2020. doi: 10.1007/s10664-020-09818-7.

M. Felderer and R. Ramler. Integrating risk-based testing in industrial test processes. *Software Quality Journal*, 22(3):543–575, 2014. doi: 10.1007/s11219-013-9226-y.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

M. Felderer and I. Schieferdecker. A taxonomy of risk-based testing. *International Journal on Software Tools for Technology Transfer*, 16:559–568, 10 2014. doi: 10.1007/s10009-014-0332-3.

J. Grossmann, M. Felderer, J. Viehmann, and I. Schieferdecker. A taxonomy to assess and tailor risk-based testing in recent testing standards. *IEEE Software*, 37(1):40–49, 2020. doi: 10.1109/MS.2019.2915297.

A. Grzymek. Research information system for a smart lab use case scaling up of biofuel plant models. Master's thesis, Vienna University of Technology, 2022. URL https://repositum.tuwien.at/handle/20.500.12708/20134.

R. Hametner, D. Winkler, and A. Zoitl. Agile testing concepts based on keyword-driven testing for industrial automation systems. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 3727–3732, 2012. doi: 10.1109/IECON.2012.6389298.

P. Hopkin. *Fundamentals of Risk Management: Understanding, Evaluating and Implementing Effective Risk Management*. Kogan Page, 5th edition, 2018.

W. S. Humphrey. The software engineering process: Definition and scope. *SIGSOFT Softw. Eng. Notes*, 14(4):82–83, Apr. 1988. ISSN 0163-5948. doi: 10.1145/75111.75122. URL https://doi.org/10.1145/75111.75122.

L. Hundt and A. Lüder. Development of a method for the implementation of interoperable tool chains applying mechatronical thinking — use case engineering of logic control. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, pages 1–8, 2012. doi: 10.1109/ETFA.2012.6489661.

ISO25030. Iso/iec 25030:2019. systems and software engineering — systems and software quality requirements and evaluation (square) — quality requirements framework, 2019. https://www.iso.org/standard/72116.html; (Accessed on 2022-09-25).

ISO25040. Iso/iec 25040:2011. systems and software engineering — systems and software quality requirements and evaluation (square) — evaluation process, 2011. https://www.iso.org/standard/35765.html; (Accessed on 2021-06-16).

J. M. Juran and J. F. Riley. *The quality improvement process*. McGraw Hill New York, 1999.

T. Jäger, A. Fay, T. Wagner, and U. Löwen. Mining technical dependencies throughout engineering process knowledge. In *ETFA2011*, pages 1–7, 2011. doi: 10.1109/ETFA.2011.6058985.

H. Kagermann, W. Wahlster, and J. Helbig. Recommendations for implementing the strategic initiative industrie 4.0 – securing the future of german manufacturing industry. Final report of the industrie 4.0 working group, acatech – National Academy of Science

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

and Engineering, München, apr 2013. URL http://forschungsunion.de/pdf/industrie_4_0_final_report.pdf.

B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering, technical report ebse 2007-001. 2007.

S. Kropatschek, O. Gert, I. Ayatollahi, K. Meixner, E. Kiesling, A. Steigberger, A. Luder, and S. Biffl. Designing a digital shadow for efficient, low-delay analysis of production quality risk. In *2022 27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Stuttgart, Germany, 2022. 8p.

D. R. Kuhn, R. N. Kacker, and Y. Lei. Sp 800-142. practical combinatorial testing. Technical report, Gaithersburg, MD, USA, 2010.

Z. Li and R. Kang. Strategy for reliability testing and evaluation of cyber physical systems. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1001–1006, 2015. doi: 10.1109/IEEM.2015.7385799.

J. Meier, H. Klare, C. Tunjic, C. Atkinson, E. Burger, R. Reussner, and A. Winter. Single underlying models for projectional, multi-view environments. pages 119–130, 01 2019. doi: 10.5220/0007396401190130.

J. Meier, C. Werner, K. Heiko, T. Christian, U. Aßmann, C. Atkinson, E. Burger, R. Reussner, and A. Winter. Classifying approaches for constructing single underlying models. pages 350–375, 2020.

K. Meixner, L. Kathrein, D. Winkler, A. Lüder, and S. Biffl. Efficient test case generation from product and process model properties and preconditions. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 859–866, 2020. doi: 10.1109/ETFA46521.2020.9212003.

K. Meixner, A. Luder, D. Winkler, and S. Biffl. A coordination artifact for multi-disciplinary reuse in production systems engineering. In *2022 27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Stuttgart, Germany, 2022. 8p.

M. Micallef and C. Colombo. Lessons learnt from using dsls for automated software testing. In *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 1–6, 2015. doi: 10.1109/ICSTW.2015.7107472.

G. Myers. *The Art of Software Testing*. John Wiley Sons, Inc.605 Third Ave. New York, NYUnited States, 1979. ISBN 978-0-471-04328-7.

E. Note Narciso, M. Delamaro, and F. Nunes. Test case selection: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering*, 24: 653–676, 05 2014. doi: 10.1142/S0218194014500259.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

J. Pfrommer, D. Štogl, K. Aleksandrov, S. Navarro, B. Hein, and J. Beyerer. Plug produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems. *at - Automatisierungstechnik*, 63:790–800, 10 2015. doi: 10.1515/auto-2014-1157.

A. Presley and D. H. Liles. The use of IDEF0 for the design and specification of methodologies. In *Proceedings of the 4th industrial engineering research conference*, 1995.

J. Robinson. *Likert Scale*, pages 3620–3621. Springer Netherlands, Dordrecht, 2014. ISBN 978-94-007-0753-5. doi: 10.1007/978-94-007-0753-5_1654. URL https://doi.org/10.1007/978-94-007-0753-5_1654.

P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Engg.*, 14(2):131–164, Apr. 2009. ISSN 1382-3256. doi: 10.1007/s10664-008-9102-8. URL https://doi.org/10.1007/s10664-008-9102-8.

R. Rwemalika, M. Kintis, M. Papadakis, Y. Le Traon, and P. Lorrach. On the evolution of keyword-driven test suites. In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, pages 335–345, 2019. doi: 10.1109/ICST.2019.00040.

O. Sauer, M. Schleipen, J. Jasperneite, A. Lüder, and H. Flatt. Requirements and concept for plug-and-work, 10 2015.

N. F. Schneidewind. *Computer, network, software, and hardware engineering with applications*. John Wiley & Sons, 2012.

A. Spillner, T. Linz, and H. Schaefer. *Software Testing Foundations: A Study Guide for the Certified Tester Exam*. Rocky Nook, 2014.

H. D. Stamatis. *Risk Management Using Failure Mode and Effect Analysis (FMEA)*. Quality Press, 2019.

J. Stark. Product lifecycle management. In *Product Lifecycle Management (Volume 1)*, pages 1–29. Springer, 2015.

M. Tuteja, G. Dubey, et al. A research study on importance of testing and quality assurance in software development life cycle (sdlc) models. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(3):251–257, 2012.

R. J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

D. Winkler, K. Meixner, and S. Biffl. Towards flexible and automated testing in production systems engineering projects. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 169–176, 2018. doi: 10.1109/ETFA.2018.8502650.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

D. Winkler, P. Novák, K. Meixner, J. Vyskočil, F. Rinker, and S. Biffl. Product-process-resource asset networks as foundation for improving cpps engineering. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2021. doi: 10.1109/ETFA45728.2021.9613253.

D. Winkler, S. Sherstneva, and S. Biffl. Towards multi-view test specification in cpps engineering. In *2022 27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Stuttgart, Germany, 2022. 4p.

C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2014. ISBN 3642432263.

J. Zavisa, A. Lüder, and A. Calà. Designing cooperating multi-agent systems — an extended design methodology. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pages 252–257, 2018. doi: 10.1109/ICPHYS.2018.8387668.

X. Zheng and C. Julien. Verification and validation in cyber physical systems: Research challenges and a way forward. In *2015 IEEE/ACM 1st International Workshop on Software Engineering for Smart Cyber-Physical Systems*, pages 15–18, 2015. doi: 10.1109/SEsCPS.2015.11.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.