Mälardalen University Doctoral Dissertation 438

Enhancing Industrial Requirements Processing and Reuse

Muhammad Abbas Khan

Muhammad Abbas Khan ENHANCING INDUSTRIAL REQUIREMENTS PROCESSING AND REUSE 2025

Mälardalen University

Mälardalen University Press Dissertations
No. 438

ENHANCING INDUSTRIAL REQUIREMENTS PROCESSING AND REUSE

Muhammad Abbas Khan

2025

School of Innovation, Design and Engineering

ISBN 978-91-7485-715-3
ISSN 1651-4238
Printed by E-Print AB, Stockholm, Sweden

Mälardalen University Press Dissertations
No. 438

ENHANCING INDUSTRIAL REQUIREMENTS PROCESSING AND REUSE

Muhammad Abbas Khan

Akademisk avhandling

som för avläggande av teknologie doktorsexamen i datavetenskap vid
Akademin för innovation, design och teknik kommer att offentligen försvaras
måndagen den 27 oktober 2025, 13.15 i Alfa, Mälardalens universitet, Västerås.

Fakultetsopponent: Professor Xavier Franch, Universitat Politècnica de Catalunya, Spain

Akademin för innovation, design och teknik

Abstract

We live in a world that depends on software. From the moment we log in to a banking system or when we take the bus to work, we are surrounded by software-intensive systems. These systems are often not built from scratch, but as further iterations of existing systems, adapted for different customers and market segments.

The development of such complex software and variant-intensive systems is centered around customer needs that are usually described in long documents, full of detail, and written in natural language. Companies must read through, interpret, and extract the relevant requirements, decide which teams should develop and test them, and simultaneously identify what can be reused from earlier projects. This process is often manual, carries a risk of mistakes, and demands great experience and precision.

This thesis explores how Artificial Intelligence (AI), and in particular natural language processing (NLP), can help make the process both faster and more reliable. The work is based on six scientific articles, which make four contributions, as follows. First, we study how requirements management and reuse are handled today to identify opportunities for enhancement. Next, we focus on automating the identification and allocation of requirements, so that correct requirements are identified and directed to the right teams from the start. We also develop methods for discovering which parts of previous projects can be reused, to avoid redundant development efforts. Finally, we create a pedagogical resource that enables teachers, students, and professionals to apply the technical solutions in practice.

Through these contributions, the thesis demonstrates how AI can become a powerful support in processing requirements and supporting reuse in complex software development.

ISBN 978-91-7485-715-3
ISSN 1651-4238

Sammanfattning

Vi lever i en värld som är beroende av programvara. Från det att vi loggar in på banken eller att vi tar bussen till jobbet är vi omgivna av programvaruintensiva system. Ofta byggs dessa system inte från grunden, utan som vidareutvecklingar av redan befintliga lösningar, anpassade för olika kunder och marknader.

Kundernas behov beskrivs vanligen i långa dokument, fulla av detaljer och skrivna på vanligt språk. Företagen måste läsa igenom, tolka och plocka ut de relevanta kraven, bestämma vilka team som ska utveckla och testa dem, och samtidigt se vad som kan återanvändas från tidigare projekt. Det sparar tid och pengar, men är också ett pussel som kräver stor erfarenhet och noggrannhet. I praktiken tar det ofta lång tid, innebär risk för misstag och är beroende av ett fåtal experter.

Den här avhandlingen undersöker hur artificiell intelligens (AI), och i synnerhet naturlig språkbehandling (NLP), kan hjälpa till att göra processen både snabbare och mer tillförlitlig.

Arbetet bygger på sex vetenskapliga artiklar och bidrar inom fyra områden: Först kartlägger vi hur arbetet med kravhantering och återanvändning går till idag, och var det finns störst potential till förbättring. Därefter fokuserar vi på att automatisera själva identifieringen och fördelningen av krav, så att de hamnar hos rätt team från början. Vi utvecklar också metoder för att upptäcka vilka delar av tidigare projekt som kan återanvändas, för att undvika att uppfinna hjulet på nytt. Slutligen skapar vi en pedagogisk resurs som gör det möjligt för lärare, studenter och yrkesverksamma att använda de tekniska lösningarna i praktiken.

Med hjälp av dessa insatser visar avhandlingen hur AI kan bli ett kraftfullt stöd i arbetet med att förstå, organisera och återanvända den kunskap som ryms i komplex programvaruutveckling.

Abstract

We live in a world that depends on software. From the moment we log in to a banking system or when we take the bus to work, we are surrounded by software-intensive systems. These systems are often not built from scratch, but as further iterations of existing systems, adapted for different customers and market segments.

The development of such complex software and variant-intensive systems is centered around customer needs that are usually described in long documents, full of detail, and written in natural language. Companies must read through, interpret, and extract the relevant requirements, decide which teams should develop and test them, and simultaneously identify what can be reused from earlier projects. This process is often manual, carries a risk of mistakes, and demands great experience and precision.

This thesis explores how Artificial Intelligence (AI), and in particular natural language processing (NLP), can help make the process both faster and more reliable. The work is based on six scientific articles, which make four contributions, as follows. First, we study how requirements management and reuse are handled today to identify opportunities for enhancement. Next, we focus on automating the identification and allocation of requirements, so that correct requirements are identified and directed to the right teams from the start. We also develop methods for discovering which parts of previous projects can be reused, to avoid redundant development efforts. Finally, we create a pedagogical resource that enables teachers, students, and professionals to apply the technical solutions in practice.

Through these contributions, the thesis demonstrates how AI can become a powerful support in processing requirements and supporting reuse in complex software development.

To my parents

Acknowledgments

I would like to express my heartfelt gratitude to the many people who have supported and inspired me throughout these years. First and foremost, I would like to thank my main advisor, *Prof. Daniel Sundmark*, for his kind support and constructive feedback throughout the thesis. I am also deeply grateful to my co-advisors, *Dr. Eduard Paul Enoiu*, for his guidance on study designs, and *Dr. Mehrdad Saadatmand*, for the short but incredibly effective coffee breaks that shaped much of this research, along with his kind help during my early days in Sweden. My sincere appreciation also goes to *Dr. Alessio Ferrari, Dr. Robbert Jongeling*, and *Sarmad Bashir* for their continued collaboration on various research projects.

I have been fortunate to have worked on real industrial problems at *Alstom*, made possible by the support of *Claes Lindskog, Zulqarnain Haider, and Daran Smalley*. I also wish to thank *Dr. Raluca Marinescu*, *Max Johansson, Jörgen Ekefjäll,* and all members of the Traction and TCMS software teams for their support and feedback on my work.

RISE Research Institutes of Sweden has been at the center of these collaborations with Alstom. I am especially thankful to *Larisa Rizvanovic* for her constant help and support over the years. You are truly a kind person. I often ask myself, "What would Larisa do?" when I want to be kind. I also want to thank my "academic sister", *Mahshid Helali Moghadam* (whose expertise Scania is now enjoying), for the fun talks and great after-work conversations. I would like to thank all my colleagues at RISE and my fellow Ph.D. students at Mälardalen University for their encouragement and for sharing wisdom and experiences along the way.

Last but not least, I would like to thank *Pia Olofsson* for the support during the hard times, and for pretending (very convincingly) to be interested every

time I explained my “groundbreaking” research.

The work presented in this thesis is funded by the Swedish Knowledge Foundation through the ARRAY industrial school and Vinnova through the eXcellence In Variant Testing (XIVT) and the SmartDelta projects. The work is also partially supported by KDT and Erasmus+ through the AIDOaRt and ENACTEST projects, respectively.

Muhammad Abbas Khan, Västerås, August 2025

List of Publications

Papers included in this thesis[1]

Paper A: *Muhammad Abbas*, Robbert Jongeling, Claes Lindskog, Eduard Paul Enoiu, Mehrdad Saadatmand, Daniel Sundmark. "*Product Line Adoption in Industry: An Experience Report from the Railway Domain*" In the 24th International Systems and Software Product Line Conference (SPLC 2020).

Paper B: Sarmad Bashir, *Muhammad Abbas*, Mehrdad Saadatmand, Eduard Paul Enoiu, Markus Bohlin, Pernilla Lindberg. "*Requirement or Not, That is the Question: A Case from the Railway Industry*" In the 29th International Working Conference on Requirement Engineering: Foundation for Software Quality (REFSQ 2023).

Paper C: Sarmad Bashir, *Muhammad Abbas*, Alessio Ferrari, Mehrdad Saadatmand, Pernilla Lindberg. "*Requirements Classification for Smart Allocation: A Case Study in the Railway Industry*" In the 31st International Requirements Engineering Conference (RE 2023).

Paper D: *Muhammad Abbas*, Alessio Ferrari, Anas Shatnawi, Eduard Paul Enoiu, Mehrdad Saadatmand, Daniel Sundmark. "*On the relationship between similar requirements and similar software*" In Requirements Engineering 28, 23–47 (2023).

[1] The included publications have been reformatted to comply with the thesis layout.

Paper E: *Muhammad Abbas*, Mehrdad Saadatmand, Eduard Paul Enoiu, Daniel Sundmark, Claes Lindskog. "*Automated Reuse Recommendation of Product Line Assets based on Natural Language Requirements.*" In the 19th International Conference on Software and Systems Reuse (ICSR 2020).

Paper F: *Muhammad Abbas*, Sarmad Bashir, Mehrdad Saadatmand, Eduard Paul Enoiu, Daniel Sundmark. "*Requirements Similarity and Retrieval*" In Ferrari, A., Ginde, G. (eds), Handbook on Natural Language Processing for Requirements Engineering. Springer, Cham (2025).

Contents

I

Thesis

Chapter 1

Introduction

The continuous reliance of society on software-intensive systems necessitates efficient software development processes to enable quick and quality delivery of such products. These systems are not often built from scratch, but rather as an increment or variant of an existing product platform. This approach to development avoids redundant efforts and can help improve product quality over time.

Software Product Line Engineering (SPLE) is a process that enables the engineering of product families, allowing for the customization and configuration of products as variants for various market segments [1]. SPLE is based on the reuse rationale, which enables the engineering of common domain requirements, assets, and components that can be later used and adapted in future similar products. In practice, the SPLE adoption varies with respect to reuse practices, i.e., from opportunistic reuse to engineered reuse [2]. While the former employs ad-hoc reuse practices, the latter promotes engineered reuse through systematic variation points and configurations. However, the upfront investment in engineering for reuse hinders the adoption of SPLE processes with systematic engineered reuse. Therefore, in practice, SPLE is often adopted in an evolutionary manner with *opportunistic reuse* also commonly referred to as clone-and-own reuse.

The manufacturing industry often benefits from such engineering practices. Like many other industries, the manufacturing industry (such as the railway industry) also relies on acquiring projects from external customers through a

call for tender [3]. Vendors often have to respond to a call for tender with a proposal that competes with other proposals from other vendors[1]. In such project-based setups, enabling a quick response to a call for tender and then meeting legally binding delivery timelines requires effective processing and delivery of customer requirements.

Processing requirements in such setups involve understanding and extracting technical customer requirements that can be agreed upon. Additionally, systems in such industries are highly complex and are driven by software. The safety-critical nature of the systems in the area necessitates effective software development and a rigorous system testing process. Therefore, companies often structure their teams into sub-teams responsible for the development of various system functions, as the components can be independently implemented, tested, certified for safety, and incrementally integrated [4]. In such cases, the *allocation of requirements* for development and testing also becomes crucial for project resource planning and development.

In a typical project-based SPLE setup, assets and artifacts from existing projects and products are reused and adapted in a *clone-and-own manner* to address the slightly varying requirements of a new customer. This results in companies having to maintain various versions of the systems, along with common domain assets, that cater to different customer needs in different market segments. In many cases, the product should also comply with varying regional standards and regulations, resulting in more product variants. Therefore, before starting the development of systems for a new customer, a reuse analysis must be conducted to identify reuse opportunities for existing product variants and versions, thereby avoiding redundant development efforts and reducing development lead time.

Problem. This thesis is motivated by practical problems in a project-based industrial setup at Alstom[2] in Sweden, yet relevant for many industries. In the studied context, requirements are identified from large tender documents and then assigned to various teams for development, as is common in many other industries. *We refer to requirements identification and allocation as requirements processing*. Additionally, in the studied context, the SPLE process is employed, which facilitates the opportunistic reuse of domain assets and assets

[1] We refer to such a way of working as *project-based setup.*

[2] Alstom is a French multinational rail transport systems manufacturer, available online, `https://www.alstom.com/alstom-sweden`.

from existing projects. This necessitates an analysis process to identify potential opportunities for reuse. While the current reuse analysis and requirements processing processes in the studied context have shown significant improvements in lead time reduction, they are still primarily manual and lack scalability. A fully manual requirements processing and reuse analysis makes the process dependent on the engineer's experience, is time-consuming, and can be prone to human error.

Summary of the Contributions. This thesis is focused on improving and supporting the requirements processing and reuse analysis processes in the studied project-based settings. We make four research contributions, which are documented in six peer-reviewed papers included in this thesis. In the rest of this thesis, we will refer to the contributions as 'C' followed by a number.

First, to identify the needs of our industrial partner, we document current practices and identify opportunities for enhancement in the current state of practice within the studied context (C_1). Out of many identified challenges, we initially focused on addressing the challenge of identifying reuse opportunities. While working towards supporting the reuse analysis process, we found that identifying requirements and allocating them to various teams are also two preliminary steps that are considered challenging in the studied setting. Therefore, we proposed two solutions based on natural language processing (NLP) and machine learning (ML) to identify requirements and allocate them to various teams for implementation and testing (C_2). As discussed, in a project-based setup, supporting reuse at the requirements level also becomes crucial. This thesis presents a solution for requirements-driven software reuse recommendation (C_3) that helps avoid redundant development efforts. In the process of augmenting the existing requirements processing and reuse analysis process in the studied setting, requirements similarity computation and retrieval play a crucial role. C_4 synthesizes the technical knowledge on similarity and retrieval into a pedagogical resource to support educators, students, and practitioners in applying the concepts.

Outline. This thesis is divided into two parts. Part I gives an overview of the thesis and is organized as follows. Chapter 2 provides an overview of the context and the research process followed. We briefly discuss the state-of-the-art and the preliminary concepts in Chapter 3. Chapter 4 presents the thesis contributions and the included papers. Chapter 5 concludes the thesis with a discussion on future work. Part II outlines the collection of included papers.

Chapter 2

Research Overview

In this chapter, we present the overall goals of the thesis and the research process used to realize the research goals.

2.1 Context & Research Goals

Software-intensive cyber-physical systems, such as railway vehicles and cars, often come as variants of existing products to address varying customer needs with minimal effort. When a customer requests a new product, it is often developed as an iteration or an increment over existing products. In the rail industry, a common practice is for customers to publish a call for tender, to which vendors respond with a proposal. During the acquisition phase of new projects, requirements analysts must extract, understand, and analyze the product requirements for a bid proposal. If a tender is to be granted, the supplier and vendor agree upon high-level requirements and a legally binding timeline for delivery of the product. Late deliveries of the products often result in heavy penalties [1]. Therefore, reducing product development lead time plays a crucial role in avoiding penalties and meeting the agreed-upon timeline. On the software development side of the rail industry, it is common to document a set of common high-level system functions (domain requirements) and their imple-

[1] "Alstom has already paid €80 million in fines for delaying the delivery of electric trains to Belgium" Accessed Jul 31, 2025. https://www.railway.supply/en/...

mentation to encourage reuse across projects, speed up software development, and, in turn, reduce the lead time to product development. This way of working loosely resembles what is described as SPLE in the literature [1].

In this thesis, we focus on improving the requirements processing and reuse in project-based setups— specifically in the railway industry. The main objective of the thesis is to reduce product development lead time by augmenting existing project-based software development processes with semi-automated approaches. To achieve the main objective, we first document current practices to identify challenges perceived by practitioners and opportunities for research. We aim to study current practices to guide the thesis toward practical, real-world challenges whose solutions would impact product development lead time. To this end, we define the first research goal of the thesis as follows.

> **RG$_1$:** To study the current practices, challenges, and enhancement opportunities in project-based software product development processes.

In the journey towards achieving RG$_1$, we first selected a representative industrial case, following similar project-based practices with customer requirements at the center of the development process. In particular, we chose Alstom as a representative industrial partner for this thesis. We started to study RG1 through document analysis [5], participant observation [6], and focus groups [7] with one team at Alstom. We documented the team's current work-

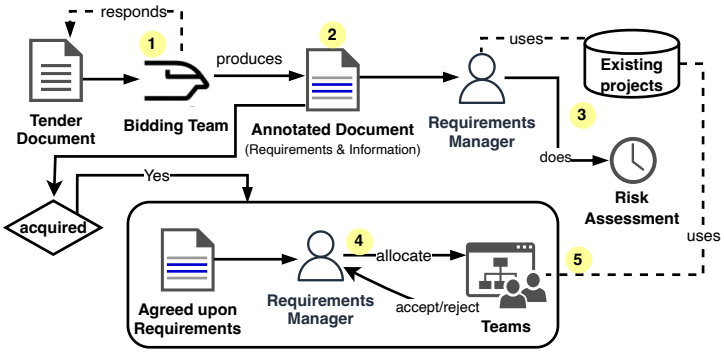

Figure 2.1: Typical flow of project acquisition and development in the studied settings

ing practices and identified several challenges that required further investigation [8].

We derive a typical overall process for the studied setting as follows. As shown in step (1) of Figure 2.1, a bidding team analyses the tender document to identify technical requirements in the large tender documents. They annotate the document to highlight potential contractual agreements, information, and requirements within the text of the document, step (2) of Figure 2.1. The annotated document is sent to the requirements engineering (RE) team for risk assessment and feasibility analysis of the project. The requirements manager relies on common requirements in the domain and existing projects to compare the project with what the company has done before. Leveraging similarities between the requirements in the tender document and the existing projects, the risk associated with the project and a tentative time plan are computed and reported to the bidding team, step (3) of Figure 2.1. The bidding team then bid for the project with a proposal based on the risk assessment. As shown in step (4) of Figure 2.1, if the company acquires the project, the agreed-upon requirements are then allocated for implementation and testing to more than twenty teams at the company. The teams can either accept the requirements or reject them based on the relevance of the requirements to their sub-systems or functions. Rejection of the allocation of requirements leads to reassignment to other teams, impacting the lead time of product development. In addition, the teams also make use of existing projects and domain assets for reuse during implementation and verification of the new project, step (5) of Figure 2.1.

From the identified challenges, we select "identifying reuse opportunities" for implementation assets to be a relevant and promising direction. Particularly when new project requirements are received, some key engineers rely on their experience to recall existing projects and domain assets and identify potential reuse opportunities. This is also seen as a challenging problem in other setups where products are derived from an existing product line [9].

While investigating solutions for semi-automated reuse identification, we found that internalized requirements documents received by technical teams at Alstom were allocated by the bidding team. The bidding team faces additional unique challenges in requirements processing. For example, identifying technical specifications in large tender documents is a preliminary step that is often manual and time-consuming, impacting project acquisition delays. In addition, if the project is acquired after bidding, the correct allocation of requirements

as work items to the technical teams plays a crucial role in reducing work allocation rejections that cause unnecessary delays.

Focusing on the above-mentioned challenges and practical problems, we defined our second research goal as reducing product development lead time by improving requirements processing and reuse identification with semi-automated approaches, as follows.

> **RG_2:** To investigate the augmentation of requirement processing and reuse in a project-based software product development environment with semi-automated approaches.

RG_2 is realized through empirical studies and semi-automated approaches at the textual requirements level, where the computation of requirements similarity plays a crucial role. Therefore, RG_2 also focuses on synthesizing technical knowledge on requirement similarity and retrieval to help practitioners apply the concepts to improve their processes in similar setups. The technical knowledge resource, a pedagogical book chapter supplemented with a code repository, is useful for educators and students in relevant fields.

2.2 Research Process

Software engineering research has struggled with practical relevance, often producing research that is difficult to apply in real-world industrial settings [10]. In addition, user perceptions and qualitative evaluations are often under-reported, limiting the understanding of how proposals perform in practice. Literature in the area recommends strong collaboration-driven co-production research processes to produce relevant research [11]. This doctoral thesis adopts a technology transfer model [12], emphasizing close collaboration between academia and industry with a strong focus on relevant problem identification, solution development & refinement, and solution deployment. The technology transfer model can be viewed as a structured extension of the broader constructive research methodology. While constructive research focuses on building and evaluating innovative artifacts to address identified problems and contribute to theoretical knowledge [13], the technology transfer model adds a practical dimension by explicitly guiding how such artifacts are developed in collaboration with industry and introduced

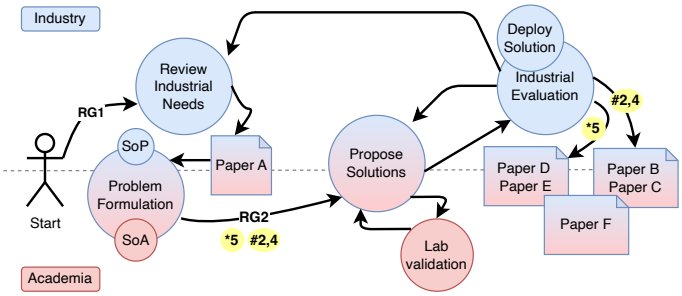

Figure 2.2: Research process followed in this thesis for technology transfer to industry. * Iteration 1 (supporting step 5 of the process presented in Figure 2.1), # Iteration 2, (supporting steps 2 and 4 of the process presented in Figure 2.1).

into real-world contexts. It extends constructive research by incorporating iterative validation and refinement steps, ensuring that the solutions are not only theoretically sound but also meet practical needs.

Beyond technical solutions, our research process also emphasizes the importance of understanding engineers' perceptions in the studied context. Through document analysis, participant observations, and focus groups, qualitative data is collected to understand and identify real-world problems (RG_1), propose relevant solutions, and assess the usability of the proposed solutions (RG_2). This approach ensures that the research outcomes are not only technically sound but also align with practitioners' needs and expectations.

We summarized each step of our collaborative research process below, shown in Figure 2.2. Figure 2.2 also maps the supported steps of Figure 2.1 and the included published papers produced from the steps of the research process.

Review of Industrial Needs. We started with RG_1 in order to review the current practices in the studied setting with the goal of identifying real challenges and practical problems. In Paper A, we started with the state-of-the-art (SoA) in similar contexts where products are developed as iterations over an existing product line to address varying customer requirements. We supplemented document analysis with around twelve months of participant observation to report the state of practice in the team developing safety-critical software systems in

the studied context. In addition, we conducted a focus group session with experts to identify key challenges and practical problems in the studied context. We recorded the focus group and performed a thematic analysis to obtain results relevant to RG_1. In the first iteration, upon agreement with the industrial partner, we focused on requirements-driven retrieval of assets for reuse, supporting step (5) of Figure 2.1. After several iterations of solution development, refinement, and validation, in the following iteration, we focused on requirements processing, supporting steps (2) and (4) of Figure 2.1.

Problem formulation. The results of Paper A inspired the initial problem formulation. One of the identified challenges (reuse identification) in Paper A was considered for further investigation in the first iteration after agreement with our industrial partner. In the second iteration, the challenges related to requirements processing were considered (requirements identification and allocation). The problems were formulated after analysis of state-of-practice (SoP) and SoA under the supervision of the researchers and the industrial partner involved. An early version of the problem related to reuse also resulted in a doctoral symposium publication [14]. This motivated the formulation of RG_2, which was realized in two iterations of the research process.

Propose Solutions & Evaluation. During the two iterations of our research process, three solutions were proposed, refined, and evaluated. In the first iteration (step *5 in Figure 2.2, which supports step 5 of Figure 2.1), we hypothesize that semantic similarity among requirements can be used to identify reuse opportunities for existing implementation assets. In Paper D, we test this hypothesis by reporting on the moderate positive association between requirements and implementation similarity. In Paper E, we proposed a solution based on requirement similarity and clustering to aid the reuse identification process in the studied context.

The execution of the first iteration of the research process led to the identification of additional challenges in requirements processing in the studied settings. In the second iteration, we focused on requirements identification (Paper B) and allocation (Paper C) to support the bidding and project management teams in the studied context.

Deploy Solution(s). The thesis resulted in three solutions over two iterations of the research process. We proposed and deployed VARA (Variability-Aware requirements Reuse Analysis), REQ-I (Requirements Identification) and REQA (smart REQuirements Allocator). VARA is deployed at the company's propulsion control and brake divisions and is frequently used in reuse identification. On the other hand, REQ-I[2] and REQA are in experimental use in the company and will be deployed in the production setup shortly.

Throughout the two iterations of our research process, requirements similarity and retrieval were central to solving practical, industry-relevant problems addressed in this thesis. This experience underscored the need for technical knowledge that is both practically grounded and accessible to educators and practitioners operating in similar contexts. To address this, Paper F closes the research process's feedback loop by consolidating the technical, methodological, and implementation insights on requirements similarity and retrieval into a pedagogical book chapter.

Practical Relevance. The proposed solutions aim to facilitate well-informed decision-making during the requirements engineering and reuse processes in project-based setups. This thesis provides a tool to support step (2) of the process in Figure 2.1, enabling requirements extraction from large tender documents with an average accuracy of 82%. This solution resulted in an estimated 80% reduction in manual efforts of requirements identification as per the evaluation conducted by Alstom in the AI-augmented Automation for DevOps, a model-based framework for continuous development in Cyber–Physical Systems (AIDOaRT) EU project [15]. Furthermore, we augment steps (4) and (5) of the process, outlined in Figure 2.1, by providing support for well-informed requirements allocation during implementation and enabling automated reuse identification. For requirements allocation, as per the company's evaluation in the AIDOaRT project, an overall 80% decrease in project efforts was seen. For reuse identification, we improved the solution and achieved an average accuracy of more than 82% in the SmartDelta project [16]. Initial estimates in the XIVT - eXcellence in Variant Testing project [17] suggest that the approach can reduce the delivery time of the propulsion sub-system by at least 20 days [3].

[2]REQ-I: https://github.com/a66as/REFSQ2023-ReqORNot

[3]ITEA News about VARA, Accessed 31 Jul, 2025. https://itea4.org/news/...

Disclaimer: The numbers and results presented in the "Practical Relevance" subsection are based on our industrial partner's estimates. The numbers are computed by a few experts who used the tools and made estimates based on experience.

Chapter 3

Background & Related Work

As mentioned in Chapter 2, the project-based setting studied in the thesis resembles the SPLE processes described in the literature. Section 3.1 covers the general SPLE process. Furthermore, the thesis's contributions build on existing algorithms and approaches from information retrieval (IR), ML, and NLP. Therefore, this chapter[1] provides a brief overview of typical text processing activities, text embeddings, and related downstream tasks in Section 3.2. Finally, Section 3.3 briefs the related work to the thesis's contributions.

3.1 Software Product Line Engineering (SPLE) in practice

SPLE is based on the rationale of engineered reuse. The SPLE development process is divided into two larger engineering phases: domain and application engineering. As shown in Figure 3.1 (inspired by Pohl *et al.* [1]), in domain engineering, experts start by scoping the domain with clearly defined requirements, which are often elicited through multiple iterations. The domain requirements are used to scope a standard architecture by allocating various common requirements to various architecture components. The standard architecture and common requirements are realized by a set of implemented reusable domain assets.

[1] This chapter borrows some concepts and parts from the licentiate thesis of the author [18].

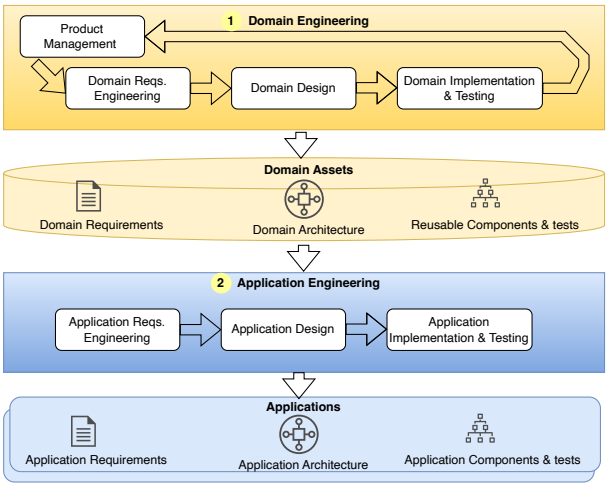

Figure 3.1: A typical software product line engineering process

When a new product (new project) is to be developed, application engineering takes over. Application engineering has its own dedicated phases for application requirements engineering, design, and implementation. The activities in application engineering often reuse existing domain assets to address the new customer's requirements. This phase could also result in "owning" the domain assets by adapting them to the new customers' requirements. Further, the phase must also consider existing application assets developed in other projects for reuse. Therefore, it is common to include reuse analysis in application engineering to find reuse opportunities across product variants (existing projects) and domain assets.

In the studied project-based development setting, SPLE is used to enable the quick and high-quality delivery of products. In such project-based setups, product-oriented thinking (like SPLE) often clashes, resulting in loosely followed SPLE processes with projects as the main focus [19]. In addition to several other organizational factors and high upfront investment in SPLE adoption, companies often opt for incremental adoption of SPLE processes with opportunistic reuse. In practice, it is common to develop domain requirements and

assets iteratively and adopt ad-hoc reuse practices [9]. This ad-hoc reuse process often relies on reusing assets across the domain and project, which leads to *cloning* existing assets and adapting them to meet new requirements. This way of reuse is known as *clone-and-own* reuse. While clone-and-own reuse is easy to implement and can result in quick product deliveries, it is unstructured and requires knowledge of existing projects and domain assets. Therefore, a reuse analysis must be done to identify existing domain and project assets that are often scattered across various repositories.

Natural language (NL) is the de facto standard for eliciting and documenting customer and domain requirements in industrial software development [20]. This widespread use extends to SPLE in practice, where organizations often rely on NL requirements to capture both commonalities and variabilities across products. However, such project-based SPLE setups frequently involve ad-hoc reuse, making the reuse and decision-making processes around requirements reuse time-consuming and error-prone. This impacts product delivery schedules. To enable efficient requirements processing and reuse, both across projects and within product lines, automation at the NL requirements level becomes essential. Such automation can reduce reliance on key experts and make the process less prone to errors.

The remainder of this chapter provides a brief overview of the background and related work in the areas related to NL requirements processing and reuse.

3.2 Text Processing & Representation

3.2.1 Pre-processing

Pre-processing of NL requirements is a preliminary step in preparing textual requirements for downstream NLP tasks. It involves changing raw and unstructured text into a clean and consistent format suitable for linguistic analysis and machine-learning approaches. This step is especially critical for requirements that often include domain-specific terms, math symbols, and inconsistent formatting.

While the specific pre-processing steps can vary based on the downstream task and the selected NLP approach, the core techniques typically include text cleaning, tokenization, part-of-speech (POS) tagging, and text normalization, as briefed below.

Text cleaning. Before analyzing requirement texts, it is important to *clean* them by removing unnecessary formatting noise and supporting information that may not be useful for automated analysis. This includes textual elements such as double spaces, tabs, punctuation, special characters, capitalization, and numbers. These parts often enhance requirements' readability and formatting, but are not helpful for NLP approaches and, in turn, can make it harder for traditional NLP approaches to understand the text correctly. An example requirement from the rail domain is shown below, both before and after cleaning.

Original: In the event that the 25kV AC supply is lost, either unexpectedly, or due to traversing a neutral section, then the ACM 400V 3 phase output shall not be interrupted while the train is moving. Any minimum operating speed that applies to this requirement shall be stated.
Cleaned: in the event that the ac supply is lost either unexpectedly or due to traversing a neutral section then the acm phase output shall not be interrupted while the train is moving any minimum operating speed that applies to this requirement shall be stated

Another important step is *tokenization*, which means breaking the text into smaller pieces like sentences or individual words. This makes the text easier to work with and helps prepare it for further analysis. For example, tokenization supports tasks like part-of-speech (POS) tagging, where each word is labeled with its grammatical role (like noun or verb), enabling text normalization by making it more consistent for machine learning or other NLP tasks. The example requirement in this case would be broken into two sentences and the following example word tokens.

[“in”, “the”, “event”, “that”, “the”, “ac”, “supply”, “is”, “lost”, ..., “moving”]

This allows for syntactic parsing and enables downstream tasks like semantic role labeling or clause identification.

POS tagging labels each word with its grammatical role, helping to distinguish components such as conditions, actions, and entities in the text. In the context of requirements, this helps distinguish between actions (verbs), entities (nouns), descriptors (adjectives), and logical connectors (conjunctions, prepositions). Accurate tagging allows for better pattern recognition in requirement statements, such as conditional phrases, system behaviors, and constraints. Further, it supports a more accurate text normalization. Below, we provide an excerpt of the POS tagging for the example requirement.

[in:IN the:DT event:NN that:IN the:DT ac:NN supply:NN is:VBZ lost:VBN either:CC unexpectedly:RB or:CC due:IN to:TO traversing:VBG a:DT neutral:JJ section:NN ...]
Key: IN: Preposition or subordinating conjunction · DT: Determiner · NN: Noun, singular or mass · VBZ: Verb, singular present · VBN: Verb, past participle · CC: Coordinating conjunction · RB: Adverb · TO: preposition or infinitive marker · VBG: Verb, present participle · JJ: Adjective

This facilitates rule-based pattern recognition (e.g., identifying condition-action structures) and supports later steps like lemmatization and entity recognition.

Text Normalization converts text into a consistent format for machine interpretation. It avoids different interpretations of the same word used in other contexts. Common approaches utilize lowercase conversion, stop-word removal, and lemmatization, as briefed below.

Lemmatization utilizes the POS tags to reduce the words of the requirements to their roots, called lemmas. This is to avoid different interpretations of the same word in different language forms. This step of *lemmatization* is crucial when analyzing requirements, as the same concept may be expressed in different grammatical forms, such as “interrupted”, “interrupts”, or “interrupting”. Converting all variants to a single, normalized form (e.g., “interrupt”) ensures semantic consistency and reduces sparsity in textual representations.

Lemmas are computed using either stemming or lemmatization algorithms. While both stemming and lemmatization aim to simplify word forms, they operate in different ways. Stemming applies crude heuristics to chop off word endings (e.g., “ed”, “ing”, “s”) without considering the syntactic correctness of the result. In contrast, lemmatization uses morphological analysis and linguistic knowledge (often via lexical databases) to reduce words to a valid lemma in the same language. For example, the Porter stemming algorithm [21] is one of the earliest and most widely used stemmers in the field, known for its speed and simplicity. However, it often leads to non-word stems (e.g., “traversing” may become “travers”). Therefore, lemmatization is generally preferred, especially in formal contexts like RE, because it preserves grammatical correctness and contextual meaning [22]. A common lemmatization approach is the WordNet lemmatizer, which performs a database lookup to determine the correct lemma based on a word’s POS tag. This ensures that different grammatical forms of a word are grouped under a common root, improving the consistency of representation in downstream tasks. Table 3.1 shows a set of words, their POS tags, the stemmed word using the Porter stemmer, and their lemmatized form using

the WordNet lemmatizer. As seen from the results in Table 3.1, lemmatization algorithms produce readable lemmas that improve consistency.

Table 3.1: Comparison of Stemming and Lemmatization on Requirement Text

Original Word	POS Tag	Stemmed (Porter)	Lemmatized (WordNet)
lost	VBN	lost	lose
interrupted	VBN	interrupt	interrupt
moving	VBG	move	move
traversing	VBG	travers	traverse
unexpectedly	RB	unexpect	unexpectedly
supply	NN	suppli	supply
neutral	JJ	neutral	neutral
POS Key	VBN: Verb, past participle · VBG: Verb, present participle · RB: Adverb · NN: Noun, singular or mass · JJ: Adjective		

Stop-word removal is used to omit words like “the”, “that”, and “shall” depending on task context. These words serve grammatical and structural purposes in a natural language but contribute less to the meaning or intent of a requirement when performing automated analysis. *Stop-word removal* can reduce the noise in data and steer automated analysis toward more informative terms.

This cleaned and normalized form is suitable for representing requirements, enabling downstream tasks (particularly, the ones based on traditional NLP) based on classification, clustering, generation, or rule-based analysis. Below, we show the text of the example requirement being replaced with the cleaned and lemmatized text for further downstream tasks.

Cleaned and normalized example requirement: event ac supply lose unexpectedly traverse neutral acm phase output interrupt train any minimum operating speed apply requirement

3.2.2 Text Representation

Textual requirements are often the primary means of documenting system needs, user expectations, and functional specifications. To enable automated analysis, reasoning, and other downstream tasks, these NL requirements must be transformed into structured, machine-interpretable forms. One common approach is to represent each requirement as a numerical vector, which

captures its linguistic and semantic properties in a form interpretable for machine learning and NLP techniques. These approaches are commonly referred to as word embedding approaches. The goal of word embedding approaches is to produce embeddings that capture both surface-level lexical features (e.g., specific terminology or term frequency) and deeper semantic relationships (e.g., intent, constraints, and dependencies) of the text. That way, the representation of similar words and requirements will also be geometrically similar, enabling more effective downstream learning tasks. Representation learning approaches range from basic lexical representations based on lexical features of the text to contextual embeddings derived from large language models (LLMs), which often capture the probabilistic distribution and latent semantic nuances. We briefly discuss some of the commonly used embedding approaches as follows.

Frequency-based representation. Due to the limited and common vocabulary used in industrial requirements [23], simple representation techniques based on word frequency and co-occurrence can sometimes yield effective results. These methods typically rely on the presence or absence of terms to construct a term-document matrix, from which numerical vectors for individual requirements are derived.

One of the most widely used approaches in this category is the Bag-of-Words (BoW) model. It generates a term-document matrix by counting how often each word appears in a requirement, disregarding word order, syntax, and grammar [24]. Despite its simplicity, BoW can be helpful when working with well-structured and domain-specific requirement texts.

Building upon BoW, the Term Frequency–Inverse Document Frequency (TFIDF) model introduces a more refined way of capturing the lexical significance of the requirement's terms. It enriches the traditional term-document matrix by weighting terms based on their importance across the entire corpus of requirements. As the name suggests, term frequencies (TF) are calculated by dividing the frequency of each term by the total number of terms in the requirements. Inverse frequencies (IDF) consider how common or rare a word is across the entire corpus (collection of requirements) and are calculated as follows.

$$\text{IDF(term)} = log(\text{\# of reqs. } / \text{ (\# of reqs. containing the term)})$$

Together, TF and IDF scores are multiplied to produce the final TFIDF score

for each term, forming a weighted term-document matrix that reflects the relevance of each term in the context of the collection of requirements.

To further enrich the representation, the term co-occurrences (n-gram) can be considered. That way, sequences of n terms could be added as single features, enriching the term-document matrix. For example, the terms "AC" and "supply" can be combined to form the bigram "AC supply", which may carry more semantic weight than the individual terms alone. However, incorporating n-grams significantly increases feature space, resulting in a high-dimensional matrix with many sparse entries (i.e., containing many zeros). To address this, dimensionality reduction approaches, such as principal component analysis (PCA) [25], are often employed. These approaches reduce the number of features while preserving the most informative features of the data, thereby enabling the construction of lower-dimensional, dense vectors that are more efficient for machine learning tasks.

Static Embeddings. Dense and semantically enriched vector representations can be generated using machine learning models designed for representation learning. Among these, neural network–based word embedding models have become prominent in recent years. This part of the subsection briefs the most commonly used models that generate fixed (static) embeddings. Such static embedding approaches associate a particular word with a single vector, which remains static regardless of the word's context in a requirement. While these embeddings do not adapt vectors to context for the same word, they can capture general semantic relationships by leveraging statistical features such as global word co-occurrence.

One of the most well-known families of static word embedding models is based on the Word2Vec architecture [26]. Word2Vec is designed to learn vector representations by capturing the local context of words within a predefined sliding window over a text corpus. This model predicts surrounding context words given a target word (or vice versa), using positive examples such as ("AC", "supply") that occur together in natural text. To improve learning and reduce overfitting, the model also generates negative examples by randomly pairing unrelated words, such as ("lost", "moving"). For vector inference, the values from the output layer serve as the vector embeddings for each word in a high-dimensional space. Doc2Vec, an extension of Word2Vec, is designed to generate embeddings for the entire document (requirement) rather than individual words [27]. It adds unique document identifiers into the training process

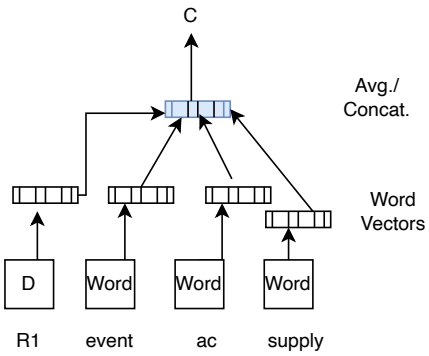

Figure 3.2: The architecture of Doc2Vec

to capture document-level semantics. The resulting document vectors are obtained by averaging or concatenating the word vectors, as shown in Figure 3.2, allowing each document to be represented as a single dense vector.

Another prominent static embedding model is GloVe (Global Vectors for Word Representation) [28], which takes a different approach by focusing on global co-occurrence statistics across the entire corpus. Instead of relying solely on local context windows, GloVe constructs a word-to-word co-occurrence matrix and uses matrix factorization techniques to learn embeddings that capture semantic relationships. For example, the vector difference between “king” and “queen” reflects latent gender distinctions. This property may not be explicitly apparent from the corpus but can be learned through global patterns.

FastText is another static embedding model developed by Facebook [29]. It builds on Word2Vec with additional subword (character-level n-grams) information, allowing the model to learn more robust representations at the character level. This subword-level modeling helps generalize better across vocabulary, especially in technical or domain-specific texts. Further, it can easily deal with out-of-vocabulary (OOV) words during inference due to its sub-word level training.

Contextual (dynamic) Embeddings. For a deeper automated understand-

ing of text, it is required to understand the context in which a words are used. For instance, the word "supply" has different contextual meanings in "power supply" and "The company supplies components...". To capture this nuance semantics, contextual embedding models are trained to produce different vector representations for the same word based on their surrounding words. We briefly present some of the notable contextual embedding approaches in the area that rely on bi-directional Long Short-Term Memory (bi-LSTM) [30] and transformer [31] architectures.

ELMo (Embeddings from Language Models) [32] is a contextual embedding model that processes input text using both forward and backward two-layer LSTMs [30]. The outputs from all layers are concatenated and reflected into a 512-dimensional dense embedding for each token. Simply put, during training, it takes the words before and after the target word as input and attempts to guess the missing word (the target). ELMo is pre-trained on a 1-billion-word corpus using language modeling objectives: *predicting the next word* in the forward direction and the previous word in the backward direction. After pretraining, only the projection layer is fine-tuned for downstream NLP tasks, allowing for transfer learning to other domains.

Another set of contextual models, trained on the same principle of next-word prediction, is based on the Transformer architecture. Notable families of models include the Bidirectional Encoder Representations from Transformers (BERT) [33], Universal Sentence Encoder (USE) [34], and Generative Pre-trained Transformer (GPT) [35]. These models are also trained in the self-supervised learning environment, often leveraging variations of the next-word prediction or masked token prediction objectives to learn contextual representations. At the core of the Transformer architecture is the attention mechanism, which enables the model to dynamically weigh the importance of different words in a sequence relative to one another. However, unlike ELMo, which processes text sequentially and is unable to capture long-range token dependencies, self-attention enables each word in a sentence to directly attend to all other words in parallel. This design enables Transformers to capture both local and global context in the text effectively. For example, in a requirement such as "If the power supply is lost unexpectedly, the system shall enter safe mode", the self-attention mechanism allows the model to learn that "lost" is semantically linked to "power supply" and that "safe mode" is a resulting action, even though these terms are separated by several tokens. Such flexible

and holistic context modeling is particularly valuable in RE, where terms can be domain-specific, technical, and context-sensitive.

BERT is designed to generate deeply bidirectional representations by considering both left and right contexts. It is trained using a masked language modeling (MLM) objective, where a certain portion of input tokens are randomly masked, and the model is tasked with predicting the original tokens. This approach enables BERT to capture nuanced contextual information from both directions.

In contrast, *GPT* follows a unidirectional (left-to-right) approach, predicting the next token in a sequence given all previous ones. This setup is effective for generative tasks such as text completion, summarization, or dialogue generation. While the original GPT model was primarily focused on language generation, later versions (e.g., GPT-4) have demonstrated strong performance in a wide range of NLP and RE tasks through few-shot or zero-shot learning without requiring task-specific fine-tuning (e.g., [36]).

Depending on the specific BERT and GPT variant and the downstream task, the requirement embedding can be derived in different ways. A common approach is to use the representation of the `[CLS]` token for BERT and the final hidden state of the last token for GPT, which captures an aggregate representation of the entire input sequence (e.g., a sentence or a requirement). Other methods for deriving embeddings include averaging all token embeddings from the final hidden layer, which is used to obtain a more evenly distributed representation of the input text.

Note that for most transformer-based models, it is often not necessary to pre-process the requirements. In addition, these models can be fine-tuned on domain-specific corpora, such as industrial requirements, to enhance performance on specialized tasks, including classification, requirement clustering, or semantic similarity.

Downstream tasks and Evaluation: Numerical multi-dimensional representation of requirements enables various downstream processing tasks. Most software and requirements engineering problems can be formulated as a downstream NLP task [37, 38], as briefed below.

- Text similarity: Duplicate or similar requirement detection is essential for reuse and quality assurance. The problem can be formulated as a text similarity problem to quantify the semantic closeness between their vector representations (e.g., [39]).

- Text clustering: The grouping of related requirements for allocation or domain requirements analysis becomes feasible through the application of clustering algorithms to embeddings (e.g., [40]).
- Recommendation and retrieval: Suggesting related requirements for reuse, linking stakeholders, retrieving relevant context, or identifying missing requirements aligns with problems in recommender systems and information retrieval, most of which rely on vector-based comparisons (e.g., [41, 42]).
- Text classification: Automatically identifying or categorizing requirements by type, quality attribute, or priority helps in the organization, traceability, and task allocation. This can be formulated as a text classification problem (e.g, [43]).
- Auto-completion and text generation: Automatically completing requirements and generating new ones based on context can be formulated as a classic text generation problem [44, 45].

To measure the effectiveness and performance of these approaches on the tasks, a variety of standard evaluation metrics are used. Similarity-based tasks often use correlation coefficients such as Pearson or Spearman rank correlation to assess how well the model's predicted similarity scores are associated with human judgment or labeled ground truth. In ranking or recommendation tasks, metrics such as Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) are commonly used to evaluate how effectively a model retrieves relevant items from a larger set. For classification tasks, precision, recall, and F1-score are widely used. Precision reflects how many of the predicted labels are correct, recall measures how many of the actual relevant items were retrieved, and the F1-score provides a harmonic mean of the two, offering a balanced view of performance. Finally, for generation tasks, the semantic alignment of the generated text and the reference text is quantified using metrics such as the BLEU (Bilingual Evaluation Understudy) score.

3.3 AI for RE and SPLE downstream tasks

The body of related work relevant to this thesis primarily explores the use of ML, NLP, artificial intelligence (AI), and IR approaches to support downstream

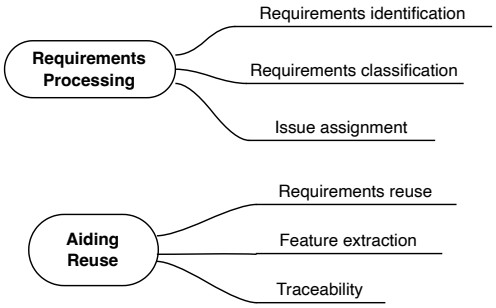

Figure 3.3: Themes in related work to this thesis

tasks in RE and SPLE. As presented in Figure 3.3, this literature can be broadly categorized into two themes: *requirements processing* and studies on *aiding reuse*.

In requirements processing, we focus on literature that often categorizes or classifies requirements to aid various RE tasks, such as allocation or identification. A closely related line of work to requirements classification is the work on automating the issue triage process. Therefore, we also include issue assignment as a related thread of work. On the other hand, studies that aid reuse include those that explore requirement reuse through automated analysis. This theme of related work sees traceability link recovery as a crucial enabler for requirements-driven software reuse. Further, another line of related work focuses on extracting and locating features to aid engineered reuse within a software product line. We provide a brief overview of the above-mentioned related areas of research as follows.

3.3.1 Requirements identification, assignment and classification

Requirements identification problem is typically formulated as a binary text classification problem in the literature. Typically, approaches in this area use various NLP and ML pipelines to classify a given text into either a requirement or information. This helps in identifying requirements in larger textual

documents, reviews, and public texts. Furthermore, work within RE also focuses on using binary classification to distinguish between functional and non-functional requirements automatically.

Abualhaija et al. [46] investigate the problem of automatically identifying and extracting requirements from large-scale tender documents, which often contain heterogeneous and unstructured text. Their approach involves constructing a feature matrix from the text of the documents and applying a range of machine-learning classification techniques to determine which segments of the text may contain potential requirements. In a comprehensive evaluation, involving 30 industrial requirements documents, their approach demonstrated a high level of effectiveness, achieving an average recall of 95%. Among the classifiers tested, a traditional Support Vector Machine (SVM)-based model consistently outperformed the others, yielding the best average performance in terms of recall. Winkler and Vogelsang [47] explore the application of deep learning for the task of requirements identification in the automotive industry. Their approach leverages word embeddings as input features to train a Convolutional Neural Network (CNN) aimed at distinguishing requirements from non-requirement text. The evaluation results indicate that the model achieved an average recall of 89%. Further analysis in their follow-up study [48] reveals that, depending on the accuracy, such automated tools can effectively reduce misclassifications, introduce fewer errors into the classification process, and accelerate the overall requirements identification process. Falkner et al. [49] address the challenge of requirements identification within the context of railway domain request-for-proposal (RFP) documents. They employ a traditional Naive Bayes (NB) classifier, trained on features extracted from the textual content of RFPs, to classify sentences as requirements or non-requirements. Their preliminary evaluation, conducted on six real-world project documents, reports an average recall of 85%, demonstrating the feasibility of lightweight machine learning approaches for requirement demarcation.

Other related work in the area often focuses on different use cases, such as identifying functional and non-functional requirements or distinguishing between privacy and usability requirements. For example, Herwanto *et al.* [50] uses the Named Entity Recognition (NER) model, trained on bi-LSTM with the conditional random field, to identify privacy requirements in user stories. More recently, Alhoshan *et al.* [51] experimented with a Zero-Shot Learning (ZSL) technique on a subset of a public dataset to distinguish between usability

and security requirements.

Requirements classification and Issue assignment. Within requirements processing, requirements allocation to various teams for implementation is an essential phase for project and resource planning. The problem of requirements allocation can be formulated as a multi-class classification problem. Work in this area often uses multi-class classification for assigning requirements to various architectural elements and classification of requirements into various sub-functional and non-functional classes. Additionally, multi-class classification is also used for assigning bugs and issues. This is a slightly different use case, but issues are also typically viewed as requirements for enhancing a product under development.

Cleland-Huang *et al.* [52, 53] uses keywords to group requirements into various non-functional classes. Their approach first identifies a set of keywords in annotated requirements and then uses this set of keywords to classify new requirements. This study introduces the classic PROMISE dataset [54], which has since become a widely adopted benchmark in RE research for evaluating classification approaches. Similar to binary classification, work in this area also typically goes beyond keywords, employing pre-processing and representing the input in the shape of feature vectors or embeddings. In this regard, Shafiq *et al.* [55] proposed the TaskAllocator that uses word embedding together with LSTM to allocate tasks in an agile setup based on roles. Their results from evaluating TaskAllocator on a public dataset show that TaskAllocator achieved an average accuracy of around 69%. Furthermore, Casamayor *et al.* [56] combines binary classification with clustering to group requirements based on functionality and to enable allocation of them to design concerns of the architecture. Their results from evaluating the approach on three sample case studies show an average accuracy of around 74%.

With the recent emergence of large language models and their applications in software engineering research [37, 57], many works have utilized transformer-based models [31] for representing and then classifying requirements using transfer learning. A widely used model is the one proposed by Devlin *et al.* [33] called BERT. Varenov *et al.* [58] utilizes a variant of the BERT language model for the representation of requirements to classify requirements into various sub-classes of security. They use a sentence-level classifier based on fine-tuned DistilBERT [59] to allocate security requirements into predefined groups. The work experiments with public datasets and shows

that the DistilBERT-based classifier achieved an average F1 score of 78%. Furthermore, a comparison by Hey *et al.* [43] compares language model-based classification with traditional approaches on publicly available datasets. They achieved an average F1 score of 87% for selected frequent classes with transfer learning. More recently, classification problems have been formulated as generative problems. In requirements classification, Alhoshan *et al.* [60] explores the use of generative LLMs for binary and multi-class classification. They compare the performance results of generative and non-generative models on three public datasets. Their results indicate that, on average, Llama performs better in the generative LLMs for classification, showing less sensitivity to dataset changes. On the other hand, non-generative models showed better performance in multi-class requirements classification tasks.

Another group of related work focuses on *issues assignment*. The focus of this group of related work is often to assign a feature request or bug report to developers or teams. For example, Aktas and Yilmaz reported their experience of automating issue assignments at a Turkish company called Softech [61]. They proposed an approach called IssueTAG, which uses the traditional TFIDF-based classification for issue assignment. Their approach achieved an accuracy of 83%, which is around 3% lower than the manual issue assignment in their context. However, due to the automation of the process, a significant reduction in issue assignment efforts has been seen. A similar work by Jonsson *et al.* [62] used an ensemble learner called Stacked Generalization (SG) on data from two different companies, achieving an accuracy of 89%. Furthermore, the work of Batista [63] uses sub-word level word embeddings for the representation of the issues as vectors. The work then uses a hierarchical classification layer based on Softmax for issue classification. Their evaluation of the approach using industrial data, in comparison with an approach from the literature, shows a 7 percent gain in accuracy.

3.3.2 Requirements reuse, feature extraction and traceability

Requirements reuse is often enabled with recommender systems. These approaches are typically based on the assumption that requirements similarity could be used as a proxy for similarity in other artifacts, such as software. Leveraging similarity in the requirements domain, the recommender systems

make cross-domain recommendations to aid various tasks within software engineering.

In the context of requirements reuse, a seminal contribution to recommender systems in RE uses TFIDF and cosine similarity to retrieve past requirements from a large industrial dataset [64]. Their tool, ReqSimile, achieved approximately 50% recall for the top 10 recommended requirements, offering significant time savings compared to traditional keyword-based searches. SoCer is another software reuse approach that focuses on recommending the reuse of Python code based on natural language input query [65]. The SoCer approach relies on an abstract syntax tree and code comments to index existing Python software that is later used to retrieve relevant code for reuse based on a user query. Recent work from Limaylla *et al.* [66] proposes the SimRE tool to recommend similar requirements in SPLE setup. They utilize and compare word embedding and sentence BERT (SBERT) models for identifying similar requirements written in Spanish. The OpenReq EU project [67, 68] offers recommendations for various stages of the RE process, including bid management, elicitation, specification, and analysis. The project employs content-based recommender systems to support various RE activities, utilizing vector-space language models for representation and similarity computation. The project's results include a service that provides means for measuring similarity between requirements using the TFIDF model.

Another group of related studies focuses on developing and structuring generic and reusable requirements tailored to aid reuse. In this regard, modeling has been employed to support this structuring, reuse, and configuration at the requirements level [69]. For example, Zen-ReqConf constructs requirements hierarchies using similarity measures, such as Jaro and Jaccard, to support automated requirements structuring and reuse in a software product line. Moon et al. [70] proposed a systematic approach known as DREAM, which derives core reusable domain requirements by analyzing legacy requirements. Further evidence of effective reuse practices is found in the work of Niu et al. [71], who combined IR and NLP with Fillmore's case grammar theory to extract functional requirement profiles and model them using the Orthogonal Variability Model (OVM). Their approach utilizes lexical affinity to extract requirements profiles, aiding in reuse. Arias et al.[72] introduced a framework for managing requirements reuse within software product lines, relying on a well-defined taxonomy to facilitate the reuse process. Similarly, a

multi-ontology-based approach [73] has been proposed to enable reuse. The domain expert is tasked with constructing domain and task-specific ontologies, while CASE tools and questionnaires are used to guide analysts in deriving and eliciting application-specific requirements, enabling reuse. Finally, catalog-based approaches have also proven valuable for reusing functional requirements across similar existing projects [74, 75, 76]. This includes mapping new project requirements into an existing catalog based on similarities to identify reusable candidates. These methods are particularly useful for small companies that do not maintain formal product line models and documentation.

Feature extraction approaches look for commonalities (similarity in artifacts), and variabilities to suggest features, extract feature diagrams, and detect redundancy or duplicates. This could be particularly useful for domain engineering and scoping product platforms. Schulze *et al.* [77] details the feature model extraction approaches that use both syntactic and semantic features to extract features and other relationships. We briefly introduce some of the approaches below.

Domain analysis can require domain knowledge and common requirements that are often extracted from publicly available documents. In this regard, online product reviews have been used for the extraction of features for reuse. Bakwr *et al.* [78] lemmatizes the input reviews and uses TFIDF to derive a term-document matrix. Based on the TFIDF scores, the approach identifies domain features. Further public documents, such as brochures, can be used to mine common domain terminologies and their variations. Ferrari *et al.* [79] use public vendor documents to identify domain-generic and domain-specific terminologies using contrastive analysis on ranked terms. These terms can help in domain analysis and feature model construction. Similarly, Zhang *et al.* [80] extracts high-level terms from requirements to support feature modeling. Their work utilizes lexical features to extract noun phrases and then adapts the TextRank algorithm to rank terms that can support domain analysts in feature modeling.

Other work in the area uses natural language requirements for system feature extraction. Sree-Kumar *et al.* [81] proposed the FeatureX framework for features and other relationship extraction to aid feature modeling. The framework utilizes lexical features to identify subject-object and noun phrases in requirements, which are then used to identify root features using TFIDF. The relationships between extracted features are then extracted using rule-based

heuristics. VIBE is another feature model extraction tool that explores ambiguity classes, such as vagueness and weakness, in requirements for detecting potential variability [82]. Together with other variability indicators, VIBE uses ambiguity indicators to extract fragments of feature models that can later be combined into a system feature model by domain experts. Maazoun*et al.* [83] compares the application of TFIDF, PCA, Latent Semantic Indexing (LSI), and clustering for feature model extraction. Their comparison considers TFIDF, PCA, and LSI for representing input requirements for clustering to extract feature models. Results show that LSI outperforms other methods in terms of recall. Fantechi *et al.* [84] explored conversational LLMs for the task of extracting variability in requirements. Their work compares the performance of LLMs in feature and variation point detection across two cases, with human judgment and other rule-based approaches for variability extraction. Evaluation shows comparable performance of the LLMs in feature and variation point detection, with the Bing LLM performing better than ChatGPT 3.5.

Traceability and feature location. Trace link recovery is closely related to requirements retrieval and plays a crucial role in enabling software reuse. This task typically involves recovering links between development artifacts, such as requirements and source code. Common approaches for trace link recovery employ the application of IR methods, such as TFIDF and LSI [85, 86, 87, 88], as well as ontology-based approaches [89, 90, 91]. IR-based methods often rely on the Vector Space Model (VSM) to compute textual similarity between artifacts, utilizing term extraction and LSI to identify potential traceability links. Ontology-based approaches provide an alternative that requires less labeled data, which can lower the barrier to adopting traceability in practice [92]. For example, Assawamekin *et al.*[91] and Mosquera *et al.*[89] proposed OntoTrace, a tracing tool that combines textual similarity and domain ontologies to enable reasoning-based trace link recovery.

With recent advances in NLP, studies have increasingly utilized LLMs for trace link recovery. For example, Hey *et al.* [93] first fine-tunes a variant of BERT for identifying parts of requirements that are irrelevant to trace link recovery. They then use the classification results as a filter for a fine-grained traceability link recovery between requirements and code. Lin *et al.* [94] proposed a BERT-based framework for traceability link recovery between code and requirements, called T-BERT. The framework utilizes a pre-trained BERT model trained on source code to train a relation classifier between

source code and requirements, incorporating additional code search data. The relation classifier is then fine-tuned on data from an open-source project for code-to-requirements traceability, enabling transfer learning. Evaluation of T-BERT in three open-source projects with three BERT architectures shows that it outperformed classic IR-based traceability approaches.

More recently, larger generative LLMs and Retrieval Augmented Generation (RAG) have also seen applications in traceability link recovery. Ge *et al.* proposed a method for trace link recovery between high and low-level requirements [95]. They fine-tuned three varying sizes of the LLaMA model with three fine-tuning strategies and compared the results with other pretrained LLMs, traditional IR-based approaches, and machine learning-based approaches for trace link recovery. Their results show that their approach outperformed traditional IR and ML-based methods for trace link recovery. Additionally, the evaluation shows that the approach outperforms the GPT4o and DeepSeekR1 models in the task of trace link recovery between high-level and low-level requirements. Hey *et al.* [96] augment pretrained LLMs with in-context retrieval for prompt creation to recover traceability links between requirements. Their analysis on six datasets shows GPT4o performs better than other open-source LLMs in terms of F1 score. Evaluation also shows that RAG combined with LLMs outperforms traditional IR-based trace link recovery approaches.

A related area of research is feature location, which aims to identify where product features are implemented in the source code, particularly useful in the context of reverse engineering or re-engineering for SPLE. These approaches often assume that feature descriptions and code share similar terms, allowing textual similarity methods to be effective. For instance, But4Reuse is a seminal reverse engineering tool that utilizes LSI to identify features across product variants and provides real-world benchmarks for evaluation of new feature relocation approaches [97]. Similarly, Zhao *et al.*[98] combine IR with branch reverse call graphs to trace features back to their implementation, filtering initial matches through graph analysis. Andam *et al.*[99] propose FLOrIDA, which utilizes TFIDF and the Apache Lucene indexing library to locate features. It ranks source files based on their similarity to feature descriptions and then uses the PageRank algorithm to prioritize likely matches for feature location.

3.4 Reflection on the Related Work

This thesis addresses gaps in requirements processing by leveraging text classification to support the identification and allocation of requirements. While existing studies have explored classification for requirements engineering, the use of transformer-based models and few-shot learning in the context of identification remains underexplored. Additionally, most research on requirements allocation relies on benchmark datasets, lacking validation in practical, industrial settings. These studies often focus solely on predicting the responsible team without providing case-based explanations to support informed allocations. In contrast, our contribution emphasizes the importance of augmenting predictions with case-based explanations to improve interpretability and support well-informed requirements allocation.

Prior work on requirements retrieval and reuse varies in scope. Although recommender systems have been applied to various requirements engineering tasks, their use in project-based environments with SPLE remains underexplored. In addition, the key underlying assumption[2] of requirements-driven retrieval for reuse recommendation largely remains untested. Moreover, there is limited qualitative evidence on the practical effectiveness of such solutions for reuse in real-world settings.

This thesis addresses these gaps by empirically validating the typical assumption that requirements similarity can be used as a proxy for software similarity in the context of retrieval for software reuse. It also gathers engineers' perspectives on the assumption and introduces a reuse recommender system to enable cross-project reuse recommendations. In addition, the thesis synthesizes technical knowledge on similarity computation methods relevant to retrieval and reuse.

[2]Reuse recommenders typically assume that requirements similarity can serve as a proxy for software similarity.

Chapter 4

Research Results

This chapter briefly recaps the thesis goals and objectives. It then presents our results and a summary of the contributions. We highlight the specific contributions of the included papers and discuss the validity of the results.

4.1 Thesis Contributions

In this thesis, we focus on improving the processing of industrial requirements and reuse in project-based setups. We first identify enhancement opportunities and then present proposals targeted at improving industrial requirements engineering processes. We found that similarity in requirements in a retrieval context plays a vital role in requirements processing and reuse. Therefore, the thesis synthesizes technical knowledge of requirements similarity and retrieval into an accessible resource, helping educators teach and practitioners apply the concepts to improve their processes. We summarize the research goals of this thesis as follows:

> *RG_1: To **study the current practices, challenges, and enhancement opportunities** in project-based software product development processes.*
> *RG_2: To **investigate the augmentation of requirement processing and reuse** in a project-based software product development environment with semi-automated approaches.*

The goals of this thesis are realized by four key contributions as follows.

Table 4.1: Mapping of contributions to the research goals

	RG_1	RG_2
C_1	X	
C_2		X
C_3	X	X
C_4	X	X

- **C_1**: Practices and Challenges:
  This initial contribution examines current practices in a project-based software development setup, documenting the perceived challenges and identifying opportunities for improvement. This contribution is essential in guiding the doctoral project towards real-world problems.

- **C_2**: Requirements Extraction and Allocation:
  Among the many identified challenges and opportunities, C_2 focuses on aiding the bidding and requirements allocation phases. This contribution is two-fold. First, it proposes an approach to extract technical requirements from tender documents, aiding the bidding phase in project-based setups. Then, after acquisition, the contribution proposes an approach for allocating the extracted and improved requirements to teams as work items.

- **C_3** Retrieval for Reuse:
  This contribution explores requirements-driven software reuse across projects. C_3 first studies— both qualitatively and quantitatively— the typical assumption of content-based recommender systems that similarity in the abstracted domain (requirements) can be used as a proxy for similarity in the detailed domain (software). C_3 then leverages this tested assumption and proposes a recommender system called VARA that recommends the reuse of software components across projects to reduce the lead time of projects. C_3 also contributes to the identification of additional enhancement opportunities for our first research goal.

- **C_4** Consolidated knowledge on similarity:
  Computing requirements similarity is central in enabling semi-automated requirements processing and reuse. C_4 puts together a

pedagogical book chapter with supplementary implementations and data to help educators, researchers, and practitioners implement the technical concepts in similar settings.

Table 4.1 presents a mapping of the contributions to our research goals. We detail the contributions as follows.

4.1.1 C1: Practices and Challenges

C_1 is mainly realized by Paper A [8] and partly by Paper D [100]. Paper A outlines the current practices of a project-based setup at a company. It presents how an evolutionary SPLE process is used to enable quick and quality delivery of new projects. Further, it presents the perceived benefits of an SPLE-based development process, its perceived challenges, areas of improvement, and the company's future vision regarding its development process. The finding shows that the company achieved significant improvements through ad-hoc manual reuse in a project-based setup. In particular, development and testing time were significantly reduced. In addition, new projects derived from domain assets experienced a confidence boost.

Several challenges and improvement opportunities were identified. The challenges were divided into three themes: Product Derivation (new project development), Automation, and SPLE Awareness. In the product derivation theme of challenges, among several concrete challenges, the identification of reuse opportunities was one of them. In the automation theme of challenges, it was observed that configuring general supporting tools for SPLE is challenging. Besides, it was also observed that the lack of SPLE awareness leads to architectural decisions that negatively impact reuse.

We initially focused on identifying reuse opportunities by retrieving similar requirements to enable reuse (C_3). While addressing the retrieval and reuse problem, we also identified more challenges related to requirements identification and allocation (C_2). Furthermore, Paper D adds additional challenges to the identification of similar requirements in the context of retrieval for reuse. In particular, Paper D highlights challenges in the standardization of component interfaces, dependencies-aware reuse, and traceability that can hinder reuse.

Validity: The papers realizing this contribution employ focus group research combined with thematic analysis of the transcripts to explore the research ques-

tions in depth. To enhance the validity of the findings, several methodological strategies were implemented to mitigate potential threats. First, the study adhered to well-established qualitative research protocols, ensuring systematic data collection and analysis. A diverse set of participants was selected from different roles and departments within the company, allowing for a broad range of perspectives and reducing the risk of bias. Further, the study design and interpretation of findings involved multiple researchers and practitioners, reducing individual bias. Beyond the focus groups, data triangulation was further achieved (in Paper A) through supplementary methods, including document analysis and participant observation. These additional sources of data provided contextual grounding and helped validate the themes identified in the focus group discussions, contributing to the overall robustness and credibility of this contribution.

Nevertheless, it is worth noting that the studies were conducted within a limited scope, focusing on a single company. While this context-specific approach restricts broad generalization, in line with the principles of case-based generalization [101], the insights derived from this research can still apply to similar organizational contexts that follow similar practices.

4.1.2 C2: Requirements Extraction and Allocation

Paper B [102] and Paper C [103] serve as the primary realizations of this thesis's contribution, both focusing on overall requirements processing. C2 is structured around two interrelated objectives: extracting requirements from large tender documents and effectively allocating these requirements to appropriate technical teams for implementation.

The first proposal, realizing this contribution, introduces an approach for extracting technical requirements from large documents. These documents, which typically define the scope, constraints, and objectives of large-scale projects, are often unstructured and written in natural language, making the task of extracting relevant technical requirements non-trivial and laborious. The proposed method leverages natural language processing techniques to identify and extract these technical requirements, enabling early-stage project acquisition activities.

The second proposal proposes an approach for allocating the extracted technical requirements to the correct implementation teams. The approach

defines requirements allocation as a classification problem and supplements classification output with case-based explanations. This smart requirements allocation approach with case-based explanations supports project managers in making well-informed allocations, ultimately improving requirements processing efficiency.

Together, these approaches enhance the overall requirements processing in the studied settings by embedding intelligence and automation in the early phases of project acquisition and help in resource planning.

Validity: The validity of the results presented in this contribution has been carefully considered from multiple dimensions to ensure the soundness of the evaluation. We formulated the problems of requirements extraction and allocation as text classification tasks, aligning with established methods in the field. Performance was assessed using widely accepted metrics such as precision, recall, F1-score, and accuracy. In the allocation task, although certain requirements in the original dataset were linked to multiple teams, the decision to restrict the model to single-team allocations was made in consultation with the industry partner. This simplification was appropriate due to the infrequency of multi-team allocations, which hinders training reliable multi-label classifiers under such a data-scarce environment.

To mitigate threats related to results credibility, model fine-tuning followed recommended practices in pipeline configuration, leveraging standard, open-source tools, and pre-trained models. Multiple configurations were explored, and their performance was evaluated using five-fold cross-validation.

The empirical evaluation was conducted using data from a single company. We don't claim a broader generalization. However, in light of case-based generalization guidelines, the findings are likely transferable to similar project-based engineering contexts, particularly within domains such as railway and automotive systems, where comparable practices and organizational structures are observed [101].

4.1.3 C3: Retrieval for Reuse

This thesis contribution is focused on supporting reuse analysis at the requirements level, addressing one of the key challenges identified in C_1. This contribution is primarily realized by Paper D [100] and Paper E [104], with Paper

F [105] offering additional technical insights related to retrieval and similarity for reuse.

To address the challenge of reuse in requirements engineering, we investigated using requirements similarity as a proxy for software similarity in the context of reuse recommendation. The underlying assumption is that software components implementing similar requirements are themselves likely to be similar and, hence, reusable. However, this assumption that "similar requirements imply similar software" has not been rigorously validated in the existing literature. To address this gap, Paper D investigates the relationship between requirements similarity and software similarity. We conducted a mixed-methods study that combines both quantitative analysis and qualitative insights from practitioners. The findings reveal a moderate positive correlation between requirements similarity and the similarity of the corresponding software implementations. Additionally, engineers expressed the intuitive belief that requirements similarity should align with software similarity in practice.

These results provide supporting empirical evidence on the fundamental assumption of content-based recommender systems in our case. In Paper E [104], we introduced a content-based recommender system to enable cross-project software reuse recommendations based on customer requirements. This approach leverages word embedding techniques combined with clustering algorithms to identify and recommend reusable domain and project assets that can be tailored to address new customer needs. Our experimental results demonstrate that the approach achieves reasonable performance in recommending relevant assets for reuse. Further, the qualitative feedback from practitioners indicates that the reuse recommendations provide valuable insights that support engineers during reuse analysis.

Validity: To ensure the robustness of the results, we validated the assumption and considered various semantic models for the recommender system. In testing the assumption, the study acknowledges the implications of using automatically generated code (a standard practice in the studied settings) for measuring software similarity, noting that this could influence the observed correlations and suggesting further validation of manually written code.

Standardized, open-source implementations were used throughout, and the study designs were reviewed by both academic researchers and industry practitioners. Practitioner involvement in qualitative data collection and in the study

designs extended to data validation and thematic analysis, adding credibility to the findings. Further, we followed standard focus group protocols to reduce potential bias.

Again, while the data originates from a single company, the results and data are likely representative of similar structured project-based engineering environments, such as railway, automotive, and aerospace. The inclusion of expert perspectives from diverse teams using different requirements engineering practices further enriches the contextual diversity of the findings. Finally, the studies ensure reliability by providing detailed discussions of the experimental setup and procedures, including replication artifacts (for papers D and F).

4.1.4 C4: Consolidated knowledge on similarity

Paper F [105] realizes this thesis contribution and is motivated by C_2, C_3, and an earlier tutorial on similarity-driven software reuse recommendation [106]. The contribution focuses on consolidating a technical knowledge resource for requirements similarity computation and retrieval. Recognizing requirements similarity as a critical enabler for a wide range of RE activities—including recommendation systems, traceability link recovery, and reuse—the contribution provides a structured and comprehensive technical resource that discusses linguistic similarity, data representation strategies, evaluation, and similarity metrics.

Further, the contribution unifies diverse NLP-based techniques—spanning from traditional lexical approaches to deep learning models—into a pedagogical book chapter accompanied by a code repository, enabling RE educators and practitioners to apply the concepts. Concrete cases, such as requirements reuse and requirements-driven software retrieval, are demonstrated, both on public and industrial data, to show the practical relevance of similarity computation pipelines.

In addition to synthesizing relevant technical knowledge on similarity, the contribution also identifies key research gaps and outlines future directions for advancing similarity computation and retrieval in RE. This work thereby serves both as a technical reference and a guide for researchers and practitioners aiming to enhance RE practices.

Validity: The validity of the results presented in this contribution is supported by the application of multiple similarity computation pipelines across real-world industrial and public cases in requirements reuse and requirements-driven software retrieval. Further, we provide comprehensive supplementary software artifacts for reproducibility.

4.2 Paper Contributions

This section presents a mapping of the included papers to the thesis contributions, the individual contributions, and brief abstracts of the included papers. Each thesis contribution is mapped to at least one included paper, as shown in Table 4.2.

Table 4.2: Mapping of contributions to the included papers

	Paper A	Paper B	Paper C	Paper D	Paper E	Paper F
C_1	X			X		
C_2		X	X			
C_3				X	X	X
C_4						X

4.2.1 Individual Contributions

I am the primary driving researcher and author for the included papers A, D, E, and F. For Paper B and Paper C, the first author and I were both co-driving researchers and authors with equal contributions. In particular, we both drove the whole research from conception to implementation. Note that other co-authors and the supervision team participated in the brainstorming and planning sessions for the research and provided useful feedback on the drafts of the included papers. Further, they also partly contributed to various supporting sections, such as related work and background.

4.2.2 Included Papers

Paper A: Product Line Adoption in Industry: An Experience Report from the Railway Domain

Authors: *Muhammad Abbas*, Robbert Jongeling, Claes Lindskog, Eduard Paul Enoiu, Mehrdad Saadatmand, Daniel Sundmark.
Published in: the 24th International Systems and Software Product Line Conference (SPLC 2020).
Abstract: The software system controlling a train is typically deployed on various hardware architectures and is required to process various signals across those deployments. Increases of such customization scenarios, as well as the needed adherence of the software to various safety standards in different application domains, has led to the adoption of product line engineering within the railway domain. This paper explores the current state-of-practice of software product line development within a team developing industrial embedded software for a train propulsion control system. Evidence is collected by means of a focus group session with several engineers and through inspection of archival data. We report several benefits and challenges experienced during product line adoption and deployment. Furthermore, we identify and discuss research opportunities, focusing in particular on the areas of product line evolution and test automation.

Paper B: Requirement or Not, That is the Question: A Case from the Railway Industry
Authors: Sarmad Bashir, *Muhammad Abbas*, Mehrdad Saadatmand, Eduard Paul Enoiu, Markus Bohlin, Pernilla Lindberg.
Published in: the 29th International Working Conference on Requirement Engineering: Foundation for Software Quality (REFSQ 2023).
Abstract: **[Context and Motivation]** Requirements in tender documents are often mixed with other supporting information. Identifying requirements in large tender documents could aid the bidding process and help estimate the risk associated with the project. **[Question/problem]** Manual identification of requirements in large documents is a resource-intensive activity that is prone to human error and limits scalability. This study compares various state-of-the-art approaches for requirements identification in an industrial context. For generalizability, we also present an evaluation on a real-world public dataset. **[Principal ideas/results]** We formulate the requirement identification problem as a binary text classification problem. Various state-of-the-art classifiers based on traditional machine learning, deep learning, and few-shot learning are evaluated for requirements identification based on accuracy, precision, re-

call, and F1 score. Results from the evaluation show that the transformer-based BERT classifier performs the best, with an average F1 score of 0.82 and 0.87 on industrial and public datasets, respectively. Our results also confirm that few-shot classifiers can achieve comparable results with an average F1 score of 0.76 on significantly lower samples, i.e., only 20% of the data. **[Contribution]** There is little empirical evidence on the use of large language models and few-shots classifiers for requirements identification. This paper fills this gap by presenting an industrial empirical evaluation of the state-of-the-art approaches for requirements identification in large tender documents. We also provide a running tool and a replication package for further experimentation to support future research in this area.

Paper C: Requirements Classification for Smart Allocation: A Case Study in the Railway Industry
Authors: Sarmad Bashir, *Muhammad Abbas*, Alessio Ferrari, Mehrdad Saadatmand, Pernilla Lindberg.
Published in: the 31st International Requirements Engineering Conference (RE 2023).
Abstract: Allocation of requirements to different teams is a typical preliminary task in large-scale system development projects. This critical activity is often performed manually and can benefit from automated requirements classification techniques. To date, limited evidence is available about the effectiveness of existing machine learning (ML) approaches for requirements classification in industrial cases. This paper aims to fill this gap by evaluating state-of-the-art language models and ML algorithms for classification in the railway industry. Since the interpretation of the results of ML systems is particularly relevant in the studied context, we also provide an information augmentation approach to complement the output of the ML-based classification. Our results show that the BERT uncased language model with the softmax classifier can allocate the requirements to different teams with a 76% F1 score when considering requirements allocation to the most frequent teams. Information augmentation provides potentially useful indications in 76% of the cases. The results confirm that currently available techniques can be applied to real-world cases, thus enabling the first step for technology transfer of automated requirements classification. The study can be useful to practitioners operating in requirements-centered contexts such as railways, where accurate requirements classification

becomes crucial for better allocation of requirements to various teams.

Paper D: On the relationship between similar requirements and similar software

Authors: *Muhammad Abbas*, Alessio Ferrari, Anas Shatnawi, Eduard Paul Enoiu, Mehrdad Saadatmand, Daniel Sundmark.

Published in: Requirements Engineering 28, 23–47 (2023).

Abstract: Recommender systems for requirements are typically built on the assumption that similar requirements can be used as proxies to retrieve similar software. When a stakeholder proposes a new requirement, natural language processing (NLP)-based similarity metrics can be exploited to retrieve existing requirements, and in turn, identify previously developed code. Several NLP approaches for similarity computation between requirements are available. However, there is little empirical evidence on their effectiveness for code retrieval. This study compares different NLP approaches, from lexical ones to semantic, deep-learning techniques, and correlates the similarity among requirements with the similarity of their associated software. The evaluation is conducted on real-world requirements from two industrial projects from a railway company. Specifically, the most similar pairs of requirements across two industrial projects are automatically identified using six language models. Then, the trace links between requirements and software are used to identify the software pairs associated with each requirements pair. The software similarity between pairs is then automatically computed with JPLag. Finally, the correlation between requirements similarity and software similarity is evaluated to see which language model shows the highest correlation and is thus more appropriate for code retrieval. In addition, we perform a focus group with members of the company to collect qualitative data. Results show a moderately positive correlation between requirements similarity and software similarity, with the pre-trained deep learning-based BERT language model with preprocessing outperforming the other models. Practitioners confirm that requirements similarity is generally regarded as a proxy for software similarity. However, they also highlight that additional aspect come into play when deciding software reuse, e.g., domain/project knowledge, information coming from test cases, and trace links. Our work is among the first ones to explore the relationship between requirements and software similarity from a quantitative and qualitative standpoint. This can be useful not only in recommender systems but

also in other requirements engineering tasks in which similarity computation is relevant, such as tracing and change impact analysis.

Paper E: Automated Reuse Recommendation of Product Line Assets based on Natural Language Requirements
Authors: *Muhammad Abbas*, Mehrdad Saadatmand, Eduard Paul Enoiu, Daniel Sundmark, Claes Lindskog
Published in: the 19th International Conference on Software and Systems Reuse (ICSR 2020).
Abstract: Software product lines (SPLs) are based on reuse rationale to aid quick and quality delivery of complex products at scale. Deriving a new product from a product line requires reuse analysis to avoid redundancy and support a high degree of asset reuse. In this paper, we propose and evaluate automated support for recommending SPL assets that can be reused to realize new customer requirements. Using the existing customer requirements as input, the approach applies natural language processing and clustering to generate reuse recommendations for unseen customer requirements in new projects. The approach is evaluated both quantitatively and qualitatively in the railway industry. Results show that our approach can recommend reuse with 74% accuracy and 57.4% exact match. The evaluation further indicates that the recommendations are relevant to engineers and can support the product derivation and feasibility analysis phase of the projects. The results encourage further study on automated reuse analysis on other levels of abstractions.

Paper F: Requirements Similarity and Retrieval
Authors: *Muhammad Abbas*, Sarmad Bashir, Mehrdad Saadatmand, Eduard Paul Enoiu, Daniel Sundmark.
Published in: Ferrari, A., Ginde, G. (eds), Handbook on Natural Language Processing for Requirements Engineering. Springer, Cham (2025).
Abstract: Requirement Engineering (RE) is crucial for identifying, analyzing, and documenting stakeholders' needs and constraints for developing software systems. In most safety-critical domains, maintaining requirements and their links to other artifacts is also often required by regulatory bodies. Furthermore, in such contexts, requirements for new products often share similarities with previous existing projects performed by the company. Therefore, similar requirements can be retrieved to facilitate the feasibility analysis of new

projects. In addition, when a new customer requests a new product, retrieval of similar requirements can enable requirements-driven software reuse and avoid redundant development efforts. Manually retrieving similar requirements for reuse is typically dependent on the engineer's experience and is not scalable, as the set could be quite large. In this regard, applying Natural Language Processing (NLP) techniques for automated similarity computation and retrieval ensures the independence of the process from the human experience and makes the process scalable. This chapter introduces linguistic similarity and several NLP-based similarity computation techniques that leverage linguistic features for similarity computation. Specifically, we cover techniques for computing similarity ranging from lexical to state-of-the-art deep neural network-based methods. We demonstrate their application in two example cases: a) requirements reuse and b) requirements-driven software retrieval. The practical guidance and example cases presented in the chapter can help practitioners apply the concepts to improve their processes where similarity computation is relevant.

Chapter 5

Conclusion & Future Work

5.1 Conclusion & Summary

The continued reliance of society on software-intensive systems necessitates the use of high-quality and efficient development processes. In domains such as rail transportation, where software plays a critical role in enabling sustainable mobility, projects often face strict delivery deadlines. Vendors failing to meet these strict delivery timelines may result in significant penalties. As a result, there is a strong demand for development approaches that support the quick and quality delivery of such systems.

To meet these demands, many organizations turn to engineering practices that prioritize systematic reuse, such as Software Product Line Engineering (SPLE). However, due to the high upfront investment of systematic reuse and SPLE adoption, companies often tend to adapt the processes to their practices, which results in ad-hoc reuse. While reusing well-defined and verified existing requirements, along with tested components, helps ensure quick and quality delivery, the process is experience-dependent and labor-intensive.

This thesis focuses on two research goals: 1) identifying enhancement opportunities in such complex processes and 2) exploring the augmentation of the existing processes with semi-automated approaches to enable quick and quality delivery of software-intensive products. The goals are realized by the six included papers, as briefly summarized below.

In Paper A, we identify *opportunities for enhancement* in a project-based

development environment within the context of SPLE adoption. The paper highlights the benefits of opportunistic reuse in the SPLE process. In particular, a significant reduction in time-to-market for the sub-system is observed. Furthermore, the opportunistic reuse allows incremental safety assessment and results in a confidence boost in the new projects. However, several challenges arise during new project development for new customers. Primarily, maintaining a high degree of asset reuse, the evolution of assets, and change impact analysis is seen to be challenging.

Further analysis of the process reveals additional challenges related to project acquisition and planning. In particular, *manual identification* and *allocation of technical requirements* from the tender documents are perceived to be challenging and might impact project acquisition and product delivery timelines. In this regard, Paper B supports project acquisition with automated tools based on large language models to automatically identify technical specifications from tender documents. As per the company's estimate, the solutions resulted in an estimated 80% reduction of manual efforts required for requirements identification. We are also looking into quality aspects of the extracted specifications [107]. Further support for project planning is achieved in Paper C with the intelligent allocation of requirements as work items to various technical teams. We fine-tune a variety of language models for requirements allocation and augment the predictions with case-based explanations to support a well-informed allocation of requirements for implementation and testing. The company's internal evaluations revealed that the requirements allocation solution could reduce the manual effort of allocating requirements by 80%.

We then focused on reducing redundant development efforts by identifying *opportunities for cross-project and platform reuse* of implementation assets based on customer requirements. We hypothesized that similarity among requirements could be used to recommend the reuse of implementation assets. In Paper D, we empirically test this hypothesis with additional qualitative insights. In particular, we applied correlation analysis to study the relationship between requirements similarity and software similarity. We found a moderate positive correlation between the two variables. Exploiting this correlation, in Paper E, we developed and evaluated a reuse recommender system. Results show that we are able to recommend the reuse of implementation assets with around 74% average accuracy. Qualitative results indicate that the reuse recom-

mendations generated by our approach are useful to engineers and can support reuse analysis activities in the studied settings. Furthermore, the company's internal estimates indicate that the approach can reduce the system's delivery time by at least 20 days.

Highlighting the importance of technical concepts in requirements similarity and retrieval for reuse, Paper F *synthesizes* the technical implementation knowledge on the topic to support further research. We demonstrate various similarity computation pipelines for requirements reuse and software retrieval on two example cases, providing a supplementary repository for replication. This knowledge resource facilitates the transfer of technical knowledge on similarity and retrieval, equipping both students and practitioners with the tools to apply these concepts effectively in real-world scenarios.

5.2 Discussion and Future Work

This section provides a brief on the possible extensions of the included papers and other future directions.

Risk assessment. While this thesis enhances the existing processes for requirements processing and reuse, it still does not target the entire process. In terms of project acquisition, the process also includes risk assessment for the upcoming project. Well-informed risk assessment processes could be enabled based on customer requirements. Novelty in new requirements in relation to existing ones can serve as an indicator for estimating risk and delivery timelines. Such an approach can utilize requirements similarity, as well as other historical indicators, such as the time required to implement similar requirements, to estimate the novelty and delivery time of the requirements. Additionally, the current backlog of the teams could be considered in the process of risk estimation to quantify and visualize the deviations from requested delivery timelines.

Requirements allocation to multiple teams and atomicity. The requirements allocation contribution of the thesis currently does not support allocating requirements to multiple teams. Hierarchical and multi-label classification approaches could be explored for this task, which involves first tagging requirements that require allocation to multiple teams and then allocating them accordingly. Further, requirements that need allocation to multiple teams may also mean that the requirements are non-atomic. Classifiers can

also help in flagging non-atomic requirements first, so that the quality can be improved [108].

Automated generation of implementation assets. Once requirements are correctly allocated, our reuse recommender enables the identification of reuse opportunities for implementation assets. The recommended implementation assets and artifacts often need to be tailored to address the new customer requirements. While question-answering support for release management is explored to guide releases [42], the adaptation of assets for future releases is yet to be automated. With recent advances in generative language models, reuse candidates can be automatically tailored for the new customer requirements. Such an approach can utilize the existing requirements, their implementation, and the new requirements as input to generate a new implementation that addresses the new customer requirement.

Multi-modality in requirements. The current approaches we use do not consider relevant information in other modalities. For example, requirements documents can contain figures and tables. Additional referenced documents may also contain architectural diagrams and other formal and informal design diagrams. However, typical automated RE approaches often remove this information during pre-processing. Approaches that consider tabular data, diagrams, implementation models, and design diagrams, along with textual requirements, can significantly enhance the representation and learning of multi-modal requirements [109]. This can enable a variety of multi-modal tasks, such as design generation based on requirements.

More evaluation in organizational and human-in-the-loop context. While automated tools have demonstrated potential to enhance the software development process, future work should focus on evaluating these tools within real-world organizational and human-in-the-loop contexts. The adoption of such approaches is often hindered not only by technical shortcomings but also by organizational culture, existing workflows, and dependency on existing legacy tool-chains. Therefore, deeper empirical studies are needed to understand how automation can be effectively integrated into existing software development processes and how it may impact human decision-making. Examining socio-technical barriers, such as tool transparency, accountability, and trust, will be crucial for aligning automated tools with organizational context.

Bibliography

[1] Klaus Pohl, Günter Böckle, and Frank Van Der Linden. *Software product line engineering: foundations, principles, and techniques*, volume 1. Springer, 2005.

[2] Niko Mäkitalo, Antero Taivalsaari, Arto Kiviluoto, Tommi Mikkonen, and Rafael Capilla. On opportunistic software reuse. *Computing*, 102(11):2385–2408, 2020.

[3] Gunnar Alexandersson and Staffan Hultén. Theory and practice of competitive tenders in passenger railway services. In *4th Conference on Railroad Industry Structure, Competition and Investment, Madrid*, pages 1–22, 2006.

[4] Thomas U Pimmler and Steven D Eppinger. Integration analysis of product decompositions. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 12822, pages 343–351. American Society of Mechanical Engineers, 1994.

[5] Glenn A Bowen et al. Document analysis as a qualitative research method. *Qualitative research journal*, 9(2):27, 2009.

[6] Barbara B Kawulich. Participant observation as a data collection method. In *Forum qualitative sozialforschung/forum: Qualitative social research*, volume 6, 2005.

[7] Jyrki Kontio, Johanna Bragge, and Laura Lehtola. The focus group method as an empirical tool in software engineering. In *Guide to advanced empirical software engineering*, pages 93–116. Springer, 2008.

[8] Muhammad Abbas, Robbert Jongeling, Claes Lindskog, Eduard Paul Enoiu, Mehrdad Saadatmand, and Daniel Sundmark. Product line adoption in industry: An experience report from the railway domain. In *Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A - Volume A*, SPLC '20, New York, NY, USA, 2020. Association for Computing Machinery.

[9] Yael Dubinsky, Julia Rubin, Thorsten Berger, Slawomir Duszynski, Martin Becker, and Krzysztof Czarnecki. An exploratory study of cloning in industrial software product lines. In *2013 17th European Conference on Software Maintenance and Reengineering*, pages 25–34. IEEE, 2013.

[10] Vahid Garousi, Markus Borg, and Markku Oivo. Practical relevance of software engineering research: synthesizing the community's voice. *Empirical Software Engineering*, 25:1687–1754, 2020.

[11] V. Basili, L. Briand, D. Bianculli, S. Nejati, F. Pastore, and M. Sabetzadeh. Software engineering research and industry: A symbiotic relationship to foster impact. *IEEE Software*, 35(05):44–49, sep 2018.

[12] Tony Gorschek, Per Garre, Stig Larsson, and Claes Wohlin. A model for technology transfer in practice. *IEEE software*, 23(6):88–95, 2006.

[13] Gordana Dodig Crnkovic. *Constructive Research and Info-computational Knowledge Generation*, pages 359–380. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[14] Muhammad Abbas. Variability aware requirements reuse analysis. In *The 42nd International Conference on Software Engineering Companion*. ACM, May 2020.

[15] Hugo Bruneliere, Vittoriano Muttillo, Romina Eramo, Luca Berardinelli, Abel Gómez, Alessandra Bagnato, Andrey Sadovykh, and Antonio Cicchetti. Aidoart: Ai-augmented automation for devops, a model-based framework for continuous development in cyber–physical systems. *Microprocessors and Microsystems*, 94:104672, 2022.

[16] Mehrdad Saadatmand, Muhammad Abbas, Eduard Paul Enoiu, Bernd-Holger Schlingloff, Wasif Afzal, Benedikt Dornauer, and Michael

Felderer. Smartdelta project: Automated quality assurance and optimization across product versions and variants. *Microprocessors and microsystems*, 103:104967, 2023.

[17] Holger Schlingloff, Peter M Kruse, and Mehrdad Saadatmand. Excellence in variant testing. In *Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems*, pages 1–2, 2020.

[18] Muhammad Abbas. *Requirements-Level Reuse Recommendation and Prioritization of Product Line Assets*. PhD thesis, Mälardalen University, 2021.

[19] Cagatay Catal. Barriers to the adoption of software product line engineering. *ACM SIGSOFT Software Engineering Notes*, 34(6):1–4, 2009.

[20] Stefan Wagner, Daniel Méndez Fernández, Michael Felderer, Antonio Vetrò, Marcos Kalinowski, Roel Wieringa, Dietmar Pfahl, Tayana Conte, Marie-Therese Christiansson, Desmond Greer, et al. Status quo in requirements engineering: A theory and a global family of surveys. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 28(2):1–48, 2019.

[21] Martin F. Porter. An algorithm for suffix stripping. *Program*, 40:211–218, 1980.

[22] Vimala Balakrishnan and Ethel Lloyd-Yemoh. Stemming and lemmatization: a comparison of retrieval performances. *Lecture Notes on Software Engineering*, 2014.

[23] A. Ferrari, G. O. Spagnolo, and S. Gnesi. Pure: A dataset of public requirements documents. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 502–505, 2017.

[24] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.

[25] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.

[26] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[27] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents, 2014.

[28] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[29] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2017.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[34] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[35] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[36] Waad Alhoshan, Alessio Ferrari, and Liping Zhao. Zero-shot learning for requirements classification: An exploratory study. *Information and Software Technology*, 159:107202, 2023.

[37] Liping Zhao, Waad Alhoshan, Alessio Ferrari, Keletso J Letsholo, Muideen A Ajagbe, Erol-Valeriu Chioasca, and Riza T Batista-Navarro. Natural language processing for requirements engineering: A systematic mapping study. *ACM Computing Surveys (CSUR)*, 54(3):1–41, 2021.

[38] Julian Frattini, Michael Unterkalmsteiner, Davide Fucci, and Daniel Mendez. Nlp4re tools: Classification, overview and management. In *Handbook on Natural Language Processing for Requirements Engineering*, pages 357–380. Springer, 2025.

[39] Per Runeson, Magnus Alexandersson, and Oskar Nyholm. Detection of duplicate defect reports using natural language processing. In *29th International Conference on Software Engineering (ICSE'07)*, pages 499–510. IEEE, 2007.

[40] José del Sagrado and Isabel M del Águila. Assisted requirements selection by clustering. *Requirements Engineering*, 26(2):167–184, 2021.

[41] Bangchao Wang, Heng Wang, Ruiqi Luo, Sen Zhang, and Qiang Zhu. A systematic mapping study of information retrieval approaches applied to requirements trace recovery. In *SEKE*, pages 1–6, 2022.

[42] Md Saleh Ibtasham, Sarmad Bashir, Muhammad Abbas, Zulqarnain Haider, Mehrdad Saadatmand, and Antonio Cicchetti. Reqrag: Enhancing software release management through retrieval-augmented llms: An industrial study. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 277–292. Springer, 2025.

[43] Tobias Hey, Jan Keim, Anne Koziolek, and Walter F Tichy. Norbert: Transfer learning for requirements classification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 169–179. IEEE, 2020.

[44] Dipeeka Luitel, Shabnam Hassani, and Mehrdad Sabetzadeh. Improving requirements completeness: Automated assistance through large language models. *Requirements Engineering*, 29(1):73–95, 2024.

[45] Xiaoli Lian, Jieping Ma, Heyang Lv, and Li Zhang. Reqcompletion: domain-enhanced automatic completion for software requirements. *Requirements Engineering*, pages 1–21, 2025.

[46] Sallam Abualhaija, Chetan Arora, Mehrdad Sabetzadeh, Lionel C Briand, and Eduardo Vaz. A machine learning-based approach for demarcating requirements in textual specifications. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 51–62. IEEE, 2019.

[47] Jonas Winkler and Andreas Vogelsang. Automatic classification of requirements based on convolutional neural networks. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 39–45. IEEE, 2016.

[48] Jonas Paul Winkler and Andreas Vogelsang. Using tools to assist identification of non-requirements in requirements specifications–a controlled experiment. In *Requirements Engineering: Foundation for Software Quality: 24th International Working Conference, REFSQ 2018, Utrecht, The Netherlands, March 19-22, 2018, Proceedings 24*, pages 57–71. Springer, 2018.

[49] Andreas Falkner, Cristina Palomares, Xavier Franch, Gottfried Schenner, Pablo Aznar, and Alexander Schoerghuber. Identifying requirements in requests for proposal: A research preview. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 176–182. Springer, 2019.

[50] Guntur Budi Herwanto, Gerald Quirchmayr, and A Min Tjoa. A named entity recognition based approach for privacy requirements engineering. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2021.

[51] Waad Alhoshan, Liping Zhao, Alessio Ferrari, and Keletso J Letsholo. A zero-shot learning approach to classifying requirements: A preliminary

study. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 52–59. Springer, 2022.

[52] Jane Cleland-Huang, Raffaella Settimi, Xuchang Zou, and Peter Solc. Automated classification of non-functional requirements. *Requirements engineering*, 12(2):103–120, 2007.

[53] Jane Cleland Huang, Raffaella Settimi, Xuchang Zou, and Peter Solc. The detection and classification of non-functional requirements with application to early aspects. In *14th IEEE International Requirements Engineering Conference (RE'06)*, pages 39–48. IEEE, 2006.

[54] Jane Cleland-Huang, Sepideh Mazrouee, Huang Liguo, and Dan Port. NFR, March 2007.

[55] Saad Shafiq, Atif Mashkoor, Christoph Mayr-Dorn, and Alexander Egyed. Taskallocator: A recommendation approach for role-based tasks allocation in agile software development. In *2021 IEEE/ACM Joint 15th International Conference on Software and System Processes (ICSSP) and 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE)*, pages 39–49. IEEE, 2021.

[56] Agustin Casamayor, Daniela Godoy, and Marcelo Campo. Functional grouping of natural language requirements for assistance in architectural software design. *Knowledge-Based Systems*, 30:78–86, 2012.

[57] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 33(8):1–79, 2024.

[58] Vasily Varenov and Aydar Gabdrahmanov. Security requirements classification into groups using nlp transformers. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 444–450. IEEE, 2021.

[59] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv:1910.01108*, 2019.

[60] Waad Alhoshan, Alessio Ferrari, and Liping Zhao. How effective are generative large language models in performing requirements classification? *arXiv preprint arXiv:2504.16768*, 2025.

[61] Ethem Utku Aktas and Cemal Yilmaz. Automated issue assignment: results and insights from an industrial case. *Empirical Software Engineering*, 25(5):3544–3589, 2020.

[62] Leif Jonsson, Markus Borg, David Broman, Kristian Sandahl, Sigrid Eldh, and Per Runeson. Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts. *Empirical Software Engineering*, 21(4):1533–1578, 2016.

[63] Arthur Batista, Fabricio D'Morison Marinho, Thiago Rocha, Wilson Oliveira Neto, Giovanni Antonaccio, Tainah Chaves, Diego Falcão, Flávia de S Santos, Felipe T Giuntini, and Juliano Efson Sales. Automated bug triaging in a global software development environment: An industry experience. In *Natural Language Processing and Information Systems: NLDB 2022, Valencia, Spain, June 15–17, 2022, Proceedings*, pages 160–171. Springer, 2022.

[64] J Natt och Dag, Björn Regnell, Vincenzo Gervasi, and Sjaak Brinkkemper. A linguistic-engineering approach to large-scale requirements management. *IEEE software*, 22(1):32–39, 2005.

[65] Md Mazharul Islam and Razib Iqbal. Socer: A new source code recommendation technique for code reuse. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1552–1557. IEEE, 2020.

[66] Maria Isabel Limaylla-Lunarejo, Nelly Condori Fernández, and Miguel Rodríguez Luaces. Simre: A requirements similarity tool for software product lines. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, pages 1473–1480, 2025.

[67] Cristina Palomares, Xavier Franch, and Davide Fucci. Personal recommendations in requirements engineering: the openreq approach. In *International working conference on requirements engineering: foundation for software quality*, pages 297–304. Springer, 2018.

[68] Alexander Felfernig, Andreas Falkner, Müslüm Atas, Xavier Franch, and Cristina Palomares. OpenReq: Recommender systems in requirements engineering. In *RS-BDA*, pages 1–4, 2017.

[69] Yan Li, Tao Yue, Shaukat Ali, and Li Zhang. Enabling automated requirements reuse and configuration. *Software and Systems Modeling*, 18(3):2177–2211, 2019.

[70] Mikyeong Moon, Keunhyuk Yeom, and Heung Seok Chae. An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. *IEEE Transactions on Software Engineering*, 31(7):551–569, 2005.

[71] Nan Niu, Juha Savolainen, Zhendong Niu, Mingzhou Jin, and Jing-ru C Cheng. A Systems Approach to Product Line Requirements Reuse. *IEEE Systems Journal*, 8:827–836, 2014.

[72] Maximiliano Arias, Agustina Buccella, and Alejandra Cechich. A Framework for Managing Requirements of Software Product Lines. *Electronic Notes in Theoretical Computer Science*, 339:5–20, 2018.

[73] Zong Yong Li, Zhi Xue Wang, Ying Ying Yang, Yue Wu, and Ying Liu. Towards a multiple ontology framework for requirements elicitation and reuse. In *Proceedings - International Computer Software and Applications Conference*, volume 1, pages 189–195, 2007.

[74] C. L. Pacheco, I. A. Garcia, J. A. Calvo-Manzano, and M. Arcilla. A proposed model for reuse of software requirements in requirements catalog. *Journal of Software: Evolution and Process*, 27(1):1–21, 2015.

[75] C. Pacheco, I. Garcia, J. A. Calvo-Manzano, and M. Arcilla. Reusing functional software requirements in small-sized software enterprises: a model oriented to the catalog of requirements. *Requirements Engineering*, 22(2):275–287, 2017.

[76] Fabiane Barreto Vavassori Benitti and Rodrigo Cezario da Silva. Evaluation of a systematic approach to requirements reuse. *Journal of Universal Computer Science*, 19(2):254–280, 2013.

[77] Sandro Schulze and Yang Li. Feature and variability extraction from natural language requirements. In *Handbook of Re-Engineering Software Intensive Systems into Software Product Lines*, pages 31–52. Springer, 2022.

[78] Noor Hasrina Bakar, Zarinah M. Kasirun, Norsaremah Salleh, and Hamid A. Jalab. Extracting features from online software reviews to aid requirements reuse. *Applied Soft Computing Journal*, 49:1297–1315, 2016.

[79] Alessio Ferrari, Giorgio O Spagnolo, and Felice Dell Orletta. Mining Commonalities and Variabilities from Natural Language Documents. In *International Software Product Line Conference*, pages 116–120, Tokyo, Japan, 2013. ACM.

[80] Jianzhang Zhang, Sisi Chen, Jinping Hua, Nan Niu, and Chuang Liu. Automatic terminology extraction and ranking for feature modeling. In *2022 IEEE 30th International Requirements Engineering Conference (RE)*, pages 51–63. IEEE, 2022.

[81] Anjali Sree-Kumar, Elena Planas, and Robert Clarisó. Extracting software product line feature models from natural language specifications. In *Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 1*, pages 43–53, 2018.

[82] Alessandro Fantechi, Stefania Gnesi, and Laura Semini. Vibe: looking for variability in ambiguous requirements. *Journal of Systems and Software*, 195:111540, 2023.

[83] Jihen Maâzoun, Hanêne Ben-Abdallah, and Nadia Bouassida. Clustering techniques for software product line feature identification. In *2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–10. IEEE, 2022.

[84] Alessandro Fantechi, Stefania Gnesi, and Laura Semini. Exploring llms' ability to detect variability in requirements. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 178–188. Springer, 2024.

[85] Jane Huffman Hayes, Alex Dekhtyar, Senthil Karthikeyan Sundaram, E. Ashlee Holbrook, Sravanthi Vadlamudi, and Alain April. REquirements TRacing On target (RETRO): Improving software maintenance through traceability recovery. *Innovations in Systems and Software Engineering*, 3(3):193–202, 2007.

[86] Johan Natt och Dag, Vincenzo Gervasi, Sjaak Brinkkemper, and Bjorn Regnell. A linguistic-engineering approach to large-scale requirements management. *IEEE Softw.*, 22(1):32–39, January 2005.

[87] Wentao Wang, Nan Niu, Hui Liu, and Zhendong Niu. Enhancing automated requirements traceability by resolving polysemy. In *Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018*, pages 40–51. IEEE, 2018.

[88] Wentao Wang, Arushi Gupta, Nan Niu, Li Da Xu, Jing Ru C. Cheng, and Zhendong Niu. Automatically Tracing Dependability Requirements via Term-Based Relevance Feedback. *IEEE Transactions on Industrial Informatics*, 14(1):342–349, 2018.

[89] David Mosquera, Marcela Ruiz, Oscar Pastor, and Jürgen Spielberger. Ontology-based automatic reasoning and nlp for tracing software requirements into models with the ontotrace tool. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 140–158. Springer, 2023.

[90] David Mosquera, Marcela Ruiz, and Oscar Pastor. Ontology-based nlp tool for tracing software requirements and conceptual models: an empirical study. *Requirements Engineering*, pages 1–29, 2025.

[91] Namfon Assawamekin, Thanwadee Sunetnanta, and Charnyote Pluempitiwiriyawej. Ontology-based multiperspective requirements traceability framework. *Knowledge and Information Systems*, 25:493–522, 2010.

[92] Marcela Ruiz, Jin Yang Hu, and Fabiano Dalpiaz. Why don't we trace? a study on the barriers to software traceability in practice. *Requirements Engineering*, 28(4):619–637, 2023.

[93] Tobias Hey, Jan Keim, and Sophie Corallo. Requirements classification for traceability link recovery. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, pages 155–167. IEEE, 2024.

[94] Jinfeng Lin, Yalin Liu, Qingkai Zeng, Meng Jiang, and Jane Cleland-Huang. Traceability transformed: Generating more accurate links with pre-trained bert models. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 324–335. IEEE, 2021.

[95] Chuyan Ge, TianTian Wang, XiaoTian Yang, and Christoph Treude. Cross-level requirements tracing based on large language models. *IEEE Transactions on Software Engineering*, 2025.

[96] Tobias Hey, Dominik Fuchß, Jan Keim, and Anne Koziolek. Requirements traceability link recovery via retrieval-augmented generation. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 381–397. Springer, 2025.

[97] Jabier Martinez, Tewfik Ziadi, Mike Papadakis, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. Feature location benchmark for software families using eclipse community releases. In *International Conference on Software Reuse*, pages 267–283. Springer, 2016.

[98] Wei Zhao, Lu Zhang, Yin Liu, Jiasu Sun, and Fuqing Yang. Sniafl: Towards a static noninteractive approach to feature location. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15(2):195–226, 2006.

[99] Berima Andam, Andreas Burger, Thorsten Berger, and Michel RV Chaudron. Florida: Feature location dashboard for extracting and visualizing feature traces. In *Proceedings of the Eleventh International Workshop on Variability Modelling of Software-intensive Systems*, pages 100–107, 2017.

[100] Muhammad Abbas, Alessio Ferrari, Anas Shatnawi, Eduard Enoiu, Mehrdad Saadatmand, and Daniel Sundmark. On the relationship between similar requirements and similar software: A case study in the railway domain. *Requirements Engineering*, 28(1):23–47, 2023.

[101] Roel Wieringa and Maya Daneva. Six strategies for generalizing software engineering theories. *Science of computer programming*, 101:136–152, 2015.

[102] Sarmad Bashir, Muhammad Abbas, Mehrdad Saadatmand, Eduard Paul Enoiu, Markus Bohlin, and Pernilla Lindberg. Requirement or not, that is the question: A case from the railway industry. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 105–121. Springer, 2023.

[103] Sarmad Bashir, Muhammad Abbas, Alessio Ferrari, Mehrdad Saadatmand, and Pernilla Lindberg. Requirements classification for smart allocation: A case study in the railway industry. In *2023 IEEE 31st International Requirements Engineering Conference (RE)*, pages 201–211. IEEE, 2023.

[104] Muhammad Abbas, Mehrdad Saadatmand, Eduard Enoiu, Daniel Sundamark, and Claes Lindskog. Automated reuse recommendation of product line assets based on natural language requirements. In Sihem Ben Sassi, Stéphane Ducasse, and Hafedh Mili, editors, *Reuse in Emerging Software Engineering Practices*, pages 173–189, Cham, 2020. Springer International Publishing.

[105] Muhammad Abbas, Sarmad Bashir, Mehrdad Saadatmand, Eduard Paul Enoiu, and Daniel Sundmark. Requirements similarity and retrieval. In *Handbook on Natural Language Processing for Requirements Engineering*, pages 61–88. Springer, 2025.

[106] Muhammad Abbas, Mehrdad Saadatmand, and Eduard Paul Enoiu. Requirements-driven reuse recommendation. In *Proceedings of the 25th ACM International Systems and Software Product Line Conference-Volume A*, pages 210–210, 2021.

[107] Sarmad Bashir, Alessio Ferrari, Abbas Khan, Per Erik Strandberg, Zulqarnain Haider, Mehrdad Saadatmand, and Markus Bohlin. Requirements ambiguity detection and explanation with llms: An industrial study. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2025.

[108] Fahrizal Halim and Daniel Siahaan. Detecting non-atomic requirements in software requirements specifications using classification methods. In *2019 1st International Conference on Cybernetics and Intelligent System (ICORIS)*, volume 1, pages 269–273. IEEE, 2019.

[109] Fanyu Wang, Chetan Arora, Yonghui Liu, Kaicheng Huang, Chakkrit Tantithamthavorn, Aldeida Aleti, Dishan Sambathkumar, and David Lo. Multi-modal requirements data-based acceptance criteria generation using llms. *arXiv preprint arXiv:2508.06888*, 2025.

ISBN 978-91-7485-715-3
ISSN 1651-4238

Address:
P.O. Box 883, SE-721 23 Västerås. Sweden
P.O. Box 325, SE-631 05 Eskilstuna. Sweden
E-mail: info@mdu.se **Web:** www.mdu.se

Mälardalen University