POLITECNICO DI TORINO

MSc in Engineering and Management

Enhancing Engineering Requirements Traceability through AI: Development and Validation of a Semantic Tool

Supervisors

Filippo Maria Ottaviani

Alberto De Marco

Candidate

Alberto Bambacigno

November 2025

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern engineering projects generate thousands of requirements across multiple abstraction levels, from stakeholder needs to detailed component specifications. Managing these requirements consistently is a critical challenge, especially when they are stored and maintained in large Requirements Management (RM) databases such as Siemens Polarion . A central aspect of requirements management is traceability, that is the ability to define and maintain logical links between requirements and other engineering artifacts. Although most RM tools allow engineers to create parent–child relationships, the process of identifying the correct father requirement (the higher-level requirement that provides the motivation and rationale for a newly written one) remains usually manual, time-consuming, and error-prone. In practice, engineers must search through extensive repositories of requirements to identify suitable parent requirements. This increases the likelihood of creating orphan requirements, i.e. specifications that lack a clear justification in the hierarchy, and ultimately weakens the integrity of the entire requirement set.

1.1 Problem Existence

The development of complex technical systems requires structured approaches to coordinate the interaction of multiple disciplines, processes, and stakeholders. Systems Engineering (SE) has emerged as the discipline that provides such structure, offering methodologies and frameworks to ensure that systems meet their intended purpose, satisfy stakeholder needs, and comply with regulatory requirements. At the heart of systems engineering lies requirement engineering, the process of eliciting, analyzing, documenting, and maintaining the requirements

that define a system.

Requirements serve as the contract between stakeholders and engineers: they represent what the system should do, under what conditions, and with what constraints. In practice, requirements are not static; they evolve throughout the lifecycle of a project, from high-level stakeholder needs to detailed technical specifications for individual components. This creates an inherent need for traceability: every requirement must be linked to both its origin and its consequences, ensuring that decisions can be justified, verified, and validated.

The challenge of managing requirements is particularly evident in highly regulated and safety-critical industries, such as aerospace, defense, and automotive. In these domains, the number of requirements can reach tens of thousands, distributed across multiple abstraction levels. Each requirement may be derived from a standard, a customer request, or an internal design decision, and its fulfillment must often be demonstrated through rigorous verification and validation processes.

Within the automotive industry, and especially in the heavy truck sector, this complexity is amplified. Truck manufacturers face a combination of pressures:

- Regulatory requirements, such as emissions regulations, safety directives, and homologation processes, which impose a growing number of formalized constraints.
- Technological innovation, including electrification, connectivity, and automation, which introduces entirely new classes of requirements and interactions between subsystems.
- Customer expectations, which demand reliability, comfort, efficiency, and increasingly, digital services integrated into the vehicle.

As a result, a single truck development project may involve multiple thousands of requirements, managed across distributed teams, suppliers, and engineering domains (mechanical, electrical, software, safety).

Modern Requirements Management (RM) tools, such as Siemens Polarion have been introduced to address this challenge. These platforms provide a centralized database where requirements can be stored, linked, and traced throughout the lifecycle. They support hierarchical structures, bidirectional traceability, and compliance reporting. However, their effectiveness still depends heavily on manual effort. In particular, engineers are expected to identify the father requirement—the higher-level requirement that provides the rationale for a newly created one.

In practice, this task is extremely time-consuming and error-prone. Engineers must search

across large requirement databases, navigating through layers of abstractions and filtering through hundreds of potentially related requirements. As a result, it is common for newly created requirements to lack explicit upstream links. These orphan requirements represent a serious threat to system integrity: they exist in the database, but without a clear motivation or connection to higher-level goals.

This problem—the difficulty of establishing upstream motivational traceability in large requirement databases—exists across many industries, but is particularly pressing in automotive heavy truck development, where the scale of projects and the complexity of requirements make manual approaches unsustainable.

1.2 Problem Importance

The consequences of incomplete or inconsistent traceability extend far beyond documentation. They affect the quality, safety, cost, and schedule of engineering projects. The importance of addressing the problem of upstream requirement traceability can be understood from both technical and organizational perspectives.

1. **Technical importance**.

   From a technical standpoint, the absence of explicit links between requirements undermines the coherence of the requirement hierarchy. Systems are designed hierarchically: high-level goals are broken down into subsystem and component requirements, which in turn inform design choices and test cases. If a lower-level requirement is not clearly connected to its father requirement, it becomes unclear why the requirement exists and how it contributes to system objectives. This lack of clarity has multiple consequences:

   - **Verification and validation difficulties**: Without motivational links, it becomes harder to demonstrate that the system meets stakeholder needs or complies with standards.
   - **Design inconsistencies**: Orphan requirements may lead to features or constraints that are not aligned with system goals, creating inefficiencies or conflicts.
   - **Change management risks**: When higher-level requirements change (e.g., due to new regulations), their dependent lower-level requirements may not be updated, leading to inconsistencies and potential system failures.

   For safety-critical industries, these issues are unacceptable. Standards such as ISO 26262

in automotive or DO-178C in aerospace explicitly require end-to-end traceability, from high-level safety goals to individual verification activities. Without reliable upstream traceability, organizations cannot demonstrate compliance and face risks of certification delays, recalls, or liability.

2. **Organizational importance**.

   From an organizational perspective, the problem translates into inefficiency and cost. Engineers spend significant time manually searching through requirement repositories to identify suitable father requirements. This effort is multiplied across large teams, leading to delays in the requirements definition and review phases.

   In addition, incomplete traceability increases the likelihood of rework. When an orphan requirement is discovered late in the lifecycle, it may need to be revised, relinked, or even discarded, which disrupts project schedules and budgets. Industry studies consistently show that defects or inconsistencies discovered in later phases of development are exponentially more expensive to fix compared to those identified early.

   In the heavy truck sector, the scale of projects magnifies these costs. A development cycle may span several years and involve numerous stakeholders across different engineering domains and supplier networks. In such environments, even small inefficiencies in requirements management can accumulate into significant delays and budget overruns. At the same time, the competitive nature of the industry and the pressure to integrate emerging technologies (electrification, ADAS, connectivity) leave little room for such inefficiencies.

3. **Strategic importance**.

   Beyond technical and organizational aspects, solving the problem of upstream traceability is also a matter of strategic competitiveness. Manufacturers that can ensure complete, consistent, and efficient requirements management gain an advantage in terms of product quality, time-to-market, and compliance reliability. Conversely, those who fail to address these issues risk project delays, increased costs, and erosion of customer trust.

1.3 Old and State-of-the-Art Literature Recap

Requirements traceability has long been recognized as a cornerstone of Systems Engineering (SE), ensuring that system-level goals are systematically connected to downstream artifacts

such as design, code, and verification activities. The challenge of maintaining such traceability was first articulated by **(Gotel & Finkelstein, 1994)**, who introduced the notion of pre-requirements specification (pre-RS) traceability, stressing the difficulty of documenting the rationale behind requirements and their upstream motivations. Since then, research has addressed various strategies to create, maintain, and recover links between requirements and related artifacts.

Early approaches relied on information retrieval (IR) techniques to automatically recover traceability links between requirements and code or documentation. Recent studies have adopted machine learning and natural language processing (NLP) to enhance trace link recovery, with works such as **(Guo et al., 2025)** that demonstrate the application of deep learning models. Trace Link Recovery (TLR) approaches use techniques from information retrieval, (shallow) machine learning or deep learning. More recently, generative AI models have also been used to solve TLR by prompting LLMs. Information Retrieval A good overview of TLR approaches that use methods from information retrieval (IR) is given in the systematic mapping study conducted by Borg et al. The premise of IR-based approaches is that when two artifacts have a high degree of textual similarity, they should most likely be traced. Despite this progress, the majority of research has focused on inter-artifact traceability (requirements to design/code/tests). By contrast, intra-requirement traceability—that is, links between requirements themselves—has received comparatively less attention. Nonetheless, some literature acknowledges its importance. **(Ramesh & Jarke, 2001)** identify derivation links (a refinement relationship between requirements) and rationale links (capturing the justification of a requirement), both of which align with the concept of hierarchical "father–son" relationships.

On the industrial side, several tools have been developed to operationalize requirements traceability. Solutions such as Siemens Polarion provide broad requirements management capabilities, enabling teams to document requirements, establish traceability links, and manage change across the lifecycle. More specialized players focus on enhancing traceability and quality. For example, The REUSE Company offers semantic analysis and traceability services that integrate with platforms like Polarion, supporting consistency and compliance. Similarly, Visure Solutions delivers an Application Lifecycle Management (ALM) environment with strong emphasis on traceability, impact analysis, and compliance with safety standards such as ISO 26262.

The REUSE Company provides a suite of tools under the SES ENGINEERING Studio,

with modules such as the Requirements Quality Analyzer (RQA) and TRACEABILITY Studio. These solutions support traceability management across multiple tools, including Polarion, and rely on semantic algorithms to suggest missing or suspicious links between requirements and related artifacts. Their offering demonstrates how automation can reduce errors in traceability matrices and support engineers in managing large-scale requirement databases. However, their focus is on general trace link discovery (e.g., between regulations, requirements, design, and tests), rather than explicitly addressing the motivational hierarchy between requirements.

Visure Solutions takes a different approach, offering an integrated Application Lifecycle Management (ALM) suite with a strong emphasis on requirements traceability, compliance, and verification. Visure's platform allows users to define hierarchical relationships between requirements and perform impact analysis across the lifecycle. Some of their features incorporate AI for quality checks and compliance monitoring. Nevertheless, Visure is a full RM suite rather than a lightweight external tool, which makes it less flexible in contexts where companies already rely on established platforms like Polarion.

1.4 Gap

The review of existing literature and industrial solutions reveals a clear gap between theoretical discussions of requirement-to-requirement traceability and the practical support offered by current tools. While research has long acknowledged the importance of derivation and rationale links (Gotel and Finkelstein, 1994; Ramesh and Jarke, 2001), the majority of contributions have concentrated on connecting requirements to downstream artifacts such as design models, source code, or test cases (Antoniol et al., 2002; Guo et al., 2017). The notion of hierarchical, motivational links between requirements—where a newly formulated requirement must be explicitly justified by a higher-level "father" requirement—remains underexplored, with no automated approaches widely adopted in industry.

Siemens Polarion, widely adopted in the automotive industry, provides robust requirements management capabilities but its traceability functionalities are primarily oriented toward lifecycle artifacts and compliance assurance. The REUSE Company and Visure Solutions offer advanced features such as semantic analysis, requirement quality checks, and support for safety-critical standards (e.g., ISO 26262), yet neither directly tackles the challenge of automatically proposing appropriate hierarchical parent-child relations within a requirement set.

Lightweight add-ons, such as browser extensions, can facilitate navigation or search but stop short of offering intelligent support for establishing motivational links.

This absence is particularly critical in the automotive domain, and more specifically in the development of heavy trucks, where projects involve thousands of requirements distributed across multiple subsystems and subject to strict safety standards. In this context, requirement engineers face the recurring challenge of identifying the correct higher-level requirement when drafting new ones—a process that is currently manual, time-consuming, and error-prone.

Therefore, the gap can be summarized as follows: although theory acknowledges the necessity of intra-requirement traceability and existing tools provide strong lifecycle management and quality analysis features, no current solution effectively automates the identification of motivational father–son links within requirement hierarchies. Addressing this gap represents the core contribution of the present work.

1.5 Objectives

The main objective of this thesis is to evaluate the effectiveness and usefulness of the developed tool in supporting requirement engineers when identifying hierarchical "father" requirements. Specifically, the study aims to determine whether the tool can reduce manual effort, improve accuracy, and enhance efficiency in the analysis of motivational requirement links compared to traditional manual search methods.

To achieve this, the tool will be provided to a selected pool of requirement engineers, who will use it to establish father requirements for a set of new requirements imported from an industrial database (Polarion). The performance and utility of the tool will then be assessed through a set of key performance indicators (KPIs), including:

1. **Accuracy** – the degree to which the tool correctly suggests the appropriate father requirement.
2. **Productivity** – the reduction in time required to identify the correct father requirement compared to manual search.
3. **Usability and user satisfaction** – qualitative feedback from engineers on the tool's interface, workflow integration, and overall usefulness.

Through this evaluation, the thesis seeks to provide evidence of whether the tool can effectively support intra-requirement traceability, mitigate the challenges of manual hierarchical search,

and offer practical benefits in the context of automotive heavy-truck development projects.

1.6 Structure

This thesis is organized as follows:

- Chapter 1 introduced the existing problem and the objective of the thesis.
- Chapter 2 reviews the existing literature and tools made by competitors in a critical way.
- Chapter 3 describes the research methods.
- Chapter 4 provides the results analyzed through KPIs.
- Chapter 5 discusses the results, highlighting their theoretical and practical implications, as well as the contribution to the theory and practice.
- Chapter 6 summarizes the study, outlines its (de)limitations, and suggests possible streams of future research.

.

Chapter 2

Literature Review

A cornerstone in the academic discourse on requirements traceability is the seminal work of **(Gotel & Finkelstein, 1994)**, which remains one of the most influential and frequently cited analyses of the so-called "traceability problem." Their study is notable not only for its breadth—drawing on empirical data from over 100 practitioners, tool reviews, focus groups, and interviews—but also for the conceptual clarity it brought by distinguishing between pre-requirements specification (pre-RS) and post-requirements specification (post-RS) traceability. According to the authors, this distinction is critical for understanding why traceability has persistently remained a challenge despite decades of methodological and technological development. Post-RS traceability, which deals with the evolution of requirements once they have been codified in the requirements specification, has historically been the primary focus of both research and tool support. In contrast, pre-RS traceability, which concerns the origins of requirements, the decisions and rationales behind them, and the processes that led to their formalization, has been consistently underexplored and under-supported by existing tools.

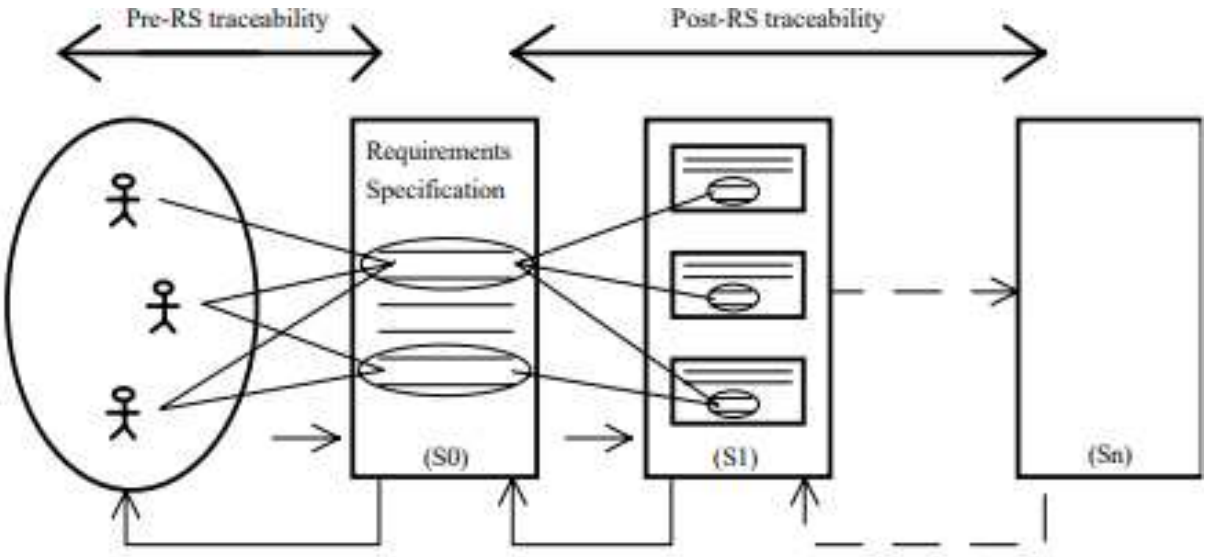

Figure 2.1: Two basic types of requirement

Gotel and Finkelstein argue that this imbalance has created a significant gap in requirements engineering practice, as many of the most pressing traceability issues originate not from the management of requirements after specification, but from the lack of structured mechanisms to capture and navigate their early justifications and relationships. A critical point in their argument is that most problems attributed to traceability failures in practice are rooted in deficiencies of pre-RS traceability. Practitioners often find themselves unable to reconstruct the rationale for why a requirement exists, who contributed to its definition, or how it relates hierarchically to other requirements. This absence of contextual information not only complicates change management but also hinders validation, reuse, and long-term maintenance of systems. The failure of existing tools to address this challenge is largely due to a reliance on rigid categorizations, manual linking schemes, and black-box treatments of the requirements specification, which obscure the inherently dynamic and social processes of requirements production. Even when post-RS traceability is well supported, the inability to map requirements back to their original sources or to the "parent" rationale from which they emerged leaves engineers with only partial visibility of the system under design.

It is precisely in this underexplored space of pre-RS traceability that the present research positions itself. The core problem addressed in this thesis—the automated retrieval of "father requirements," i.e., the higher-level requirements or rationales from which more detailed ones are derived—directly responds to the concerns articulated by Gotel and Finkelstein. Father retrieval can be understood as a mechanism to reconstruct the hierarchical and justificatory

relationships that exist among requirements, thereby enabling engineers to trace not only forwards and backwards across development artifacts but also upwards and downwards within the logical structure of the requirements set itself. This corresponds to what Gotel identified as the "invisibility" of early requirements processes: the difficulty of accessing the subtle interdependencies, trade-offs, and justifications that shaped the final specification. By addressing this problem, the tool developed in this research explicitly contributes to the area of pre-RS traceability, offering a form of support that has historically been neglected and which remains an open research challenge even in more recent literature.

Another important contribution of the Gotel and Finkelstein paper is its critical observation that the traceability problem is not uniform but multifaceted, shaped by conflicting definitions, diverse practitioner expectations, and heterogeneous project contexts. Their findings revealed that practitioners variously defined traceability as purpose-driven, solution-driven, information-driven, or direction-driven, and that no single definition encompassed all perspectives. This lack of consensus has had profound implications for tool design, resulting in fragmented approaches that fail to scale across organizational settings. The persistence of this definitional ambiguity also underscores the relevance of research like the present work, which narrows its scope to a concrete and well-defined aspect of traceability—father requirement retrieval—while acknowledging the broader conceptual debates in the field. In this way, the thesis aligns itself with the call for greater conceptual clarity and problem-specific solutions, rather than one-size-fits-all approaches that tend to collapse under the weight of diverse and conflicting expectations.

From a critical perspective, Gotel and Finkelstein's framework remains invaluable as a lens through which to analyze contemporary traceability challenges, but it also has limitations that subsequent research—including the present thesis—seeks to address. Their focus on conceptual clarity and empirical diagnosis provided a solid foundation, but their recommendations for solutions often remained at a high level of abstraction, emphasizing the need for awareness, recording, organization, and presentation of information without offering concrete mechanisms for automation or scalable integration into industrial tools. This gap is where AI-assisted approaches, such as the one developed here, can offer significant value. By leveraging natural language processing and information retrieval techniques to automate the identification of parent-child relationships between requirements, the proposed tool operationalizes the theoretical insights of Gotel and Finkelstein into a practical and testable solution. In doing so, it not only validates the enduring relevance of their diagnosis of the traceability problem but also

demonstrates how contemporary computational techniques can move the field toward actionable remedies.

In parallel with Gotel and Finkelstein's problem analysis, **(Ramesh & Jarke, 2001)** took a decisive step toward systematising the very concept of requirements traceability by proposing reference models that capture its diverse forms and uses. His contribution is particularly relevant because it moves beyond anecdotal evidence of traceability's challenges and instead builds a structured taxonomy of traceability relations grounded in empirical investigations. In his framework, traceability is not a monolithic activity but rather a set of interlinked relations that span requirements, design artefacts, rationale, and evolution paths.

Ramesh identifies different classes of traceability relations:

- Pre-requirements traceability, which connects requirements to their originating sources such as stakeholders, organisational goals, or environmental assumptions.
- Post-requirements traceability, which links requirements to downstream artefacts such as design models, code, and test cases.
- Evolution traceability, which supports understanding how requirements change over time.
- Rationale traceability, which records the motivations, justifications, and trade-offs underlying requirements.

This layered taxonomy is important for two reasons. First, it acknowledges that different projects prioritise different types of traceability (e.g., safety-critical domains often emphasise rationale and evolution, while commercial software focuses more on post-RS traceability). Second, it lays the groundwork for later attempts to operationalise traceability support in tools and methodologies.

However, a critical reading of Ramesh reveals two limitations that resonate directly with the motivation behind our research. The first is that, while the taxonomy is conceptually rich, it remains descriptive rather than prescriptive: it tells us what relations exist but not how to systematically elicit or maintain them in practice. The second is that inter-requirement links—that is, the ability to map hierarchies or dependencies between requirements themselves—are acknowledged but only superficially developed. Ramesh recognises their importance, especially for rationale traceability (e.g., capturing why requirement A exists because of requirement B), but his reference models stop short of defining concrete mechanisms for their automated identification.

This omission is significant in light of our research, since our tool specifically addresses this gap by facilitating father-child requirement retrieval. Where Ramesh provides the conceptual scaffolding to recognise such links, we aim to move one step further by testing whether AI-driven retrieval can assist engineers in uncovering them more effectively than manual methods.

In doing so, our work not only validates the enduring relevance of Ramesh's taxonomy but also demonstrates how contemporary computational techniques can operationalise areas of traceability that his models left underspecified, especially the identification of inter-requirement dependencies within large-scale repositories like Polarion.

Building on the conceptual foundations established by Ramesh, later research began to move toward operationalising traceability through automation. A particularly influential step in this direction is the work of **(Spanoudakis et al., 2004)**, who introduced a rule-based approach to automatically generate traceability relations between natural language requirements, use cases, and analysis object models. Their contribution is notable for two reasons: It represents one of the earliest attempts to bring formal structure to traceability through natural language processing (NLP), and it explicitly tackles the problem of linking heterogeneous artifacts, a challenge that continues to dominate the field.

Their framework distinguishes between two categories of rules:

1. **RTOM** (Requirements-to-Object-Model) rules – These are syntactic patterns that establish relations between fragments of requirements/use case text and elements of analysis object models. For example, they capture when a qualified noun phrase in a requirement corresponds to an attribute of a class in the object model, or when a verb phrase aligns with a stereotyped operation. The rules generate semantic relations such as Overlap (two artefacts describe the same feature) or Requires Execution Of (a requirement implies execution of a specific operation).

2. **IREQ** (Inter-Requirement) rules – These extend the framework by generating traceability links between different requirements themselves, but only indirectly. Specifically, IREQ rules infer inter-requirement relations (such as Can Partially Realise or Requires Feature In) by chaining existing RTOM relations through the shared object model. For instance, if two requirements independently reference the same class attribute or operation, the framework infers that they are semantically connected.

The strength of this approach lies in its lightweight NLP methodology. Rather than attempting

deep semantic analysis—unfeasible given the unconstrained nature of requirements text—it relies on probabilistic grammatical tagging and domain-specific heuristics to identify recurring linguistic patterns. This pragmatic balance allowed the authors to achieve relatively high levels of recall and precision in experiments on real-world datasets, outperforming contemporary information retrieval (IR) techniques such as vector space models (Antoniol et al., 2002).

Critically, however, several limitations temper the impact of Spanoudakis and Zisman's contribution. First, the approach is rule-bound and lexicon-dependent, requiring extensive manual specification of grammatical patterns and synonym sets (e.g., linking verbs like "set," "adjust," "modify" to «set» operations). This creates scalability issues: each new domain or project style of requirements writing necessitates costly tailoring of rules. Second, while IREQ rules technically address inter-requirement traceability, they do so only indirectly, relying on mediation through object models. As such, the framework never confronts head-on the challenge of identifying motivational or hierarchical requirement links—the very type of "father-child" dependency that our research addresses.

Moreover, the reliance on deterministic rules means the framework struggles with the nuances and variability of natural language. Subtle semantic cues that could imply dependency (e.g., conditional phrasing such as "in order to," or normative modal verbs like "must" vs. "should") are ignored unless explicitly encoded. This rigidity stands in contrast with modern AI methods, which, while imperfect, are able to capture patterns of usage, context, and intent that are difficult to formalise through static rules.

From the perspective of our research, the Spanoudakis and Zisman prototype marks a crucial transitional stage in the evolution of traceability research. It demonstrates the feasibility of automated support and highlights the advantages of combining linguistic analysis with structural models. At the same time, it illustrates why rule-based approaches alone are insufficient for tackling complex forms of inter-requirement traceability. Where their IREQ rules stop short—capturing only overlaps or indirect dependencies—our tool attempts to go further, by exploring whether machine learning techniques can assist requirement engineers in directly identifying father-child relations across large repositories of requirements. In this sense, our contribution can be framed as the next logical step in the trajectory: from conceptual taxonomies (Ramesh), to rule-based operationalisation (Spanoudakis and Zisman), to AI-enabled support for nuanced pre-requirements traceability.

Building on these earlier conceptual and empirical foundations, more recent work has

turned to natural language processing (NLP) as a means of operationalizing requirements traceability in contexts where requirements are largely unstructured. The chapter by **(Guo et al., 2025)** in the Handbook on Natural Language Processing for Requirements Engineering provides perhaps the most comprehensive and systematic account of this line of research to date. Their contribution is significant in at least three respects: first, it synthesizes decades of fragmented work into a coherent narrative that explicitly situates NLP within the broader goals of requirements traceability; second, it classifies the variety of NLP techniques that have been applied across different traceability tasks, from link recovery to evolution management; and third, it offers a critical reflection on the challenges that continue to limit the deployment of NLP-based traceability in industrial practice.

The authors begin by revisiting the central problem identified by both Gotel and Finkelstein and Ramesh and Jarke: the lack of effective, scalable, and sustainable support for capturing and maintaining trace links. They argue that while reference models and conceptual taxonomies (such as those proposed by Ramesh and Jarke) clarified the what of traceability, NLP provides a pathway to the how. By automating the extraction of candidate trace links from unstructured requirements texts, NLP-based methods promise to alleviate the capture burden that has historically undermined adoption. This is particularly relevant for pre-requirements traceability, which Gotel and Finkelstein had shown to be chronically neglected. Parent–child relations between requirements are often implicit, expressed in varying levels of abstraction and phrased differently by different authors. Manual identification of such relations is not only laborious but also error-prone. NLP offers a way of surfacing these hidden links by detecting semantic similarity, entailment, or hierarchical refinement cues in the text itself.

Guo et al. provide a detailed taxonomy of NLP techniques employed in this domain, ranging from traditional information retrieval (IR) methods, such as vector space models and TF–IDF weighting, to more advanced approaches based on deep learning and contextual embeddings (e.g., BERT, RoBERTa). They note that early IR-based methods achieved reasonable recall but often suffered from poor precision, as they lacked the ability to capture semantic nuance beyond keyword overlap. This criticism resonates with the findings of Spanoudakis et al. (2004), whose rule-based approach attempted to go beyond surface similarity by encoding grammatical patterns. However, as Guo et al. point out, rule-based methods tend to be brittle and require significant expert input to generalize across domains. Modern embedding-based approaches, by contrast, can capture latent semantic relationships and therefore offer greater

potential for identifying parent–child requirements even when they share little surface vocabulary.

Yet, the chapter is far from an uncritical celebration of NLP. Guo and colleagues emphasize that despite technical progress, the deployment of NLP-based traceability in industry remains limited. A central challenge is domain adaptation: models trained on generic corpora may fail to capture the technical semantics of requirements in specialized domains such as automotive, aerospace, or healthcare. Without careful fine-tuning on domain-specific data, even state-of-the-art embeddings can produce misleading similarities. Furthermore, NLP models tend to treat traceability as a pairwise classification problem (does requirement A trace to requirement B?), but this framing overlooks the hierarchical and network-like structure of real-world traceability, where links are not binary but embedded in a system of dependencies, rationales, and constraints. In this sense, the chapter implicitly echoes Ramesh and Jarke's earlier insistence on the semantics of trace links: automation is useful only if it respects the meaning and intended use of the relations it creates.

For our purposes, the critical relevance of Guo et al.'s account lies in how it bridges the historical gap between the conceptual aspirations of traceability research and the computational means to realize them. Their survey confirms that the particular kind of relation we are targeting — the identification of parent (or "father") requirements — is not only underexplored but also well-suited to NLP-based support. Unlike cross-artifact traceability (e.g., requirement-to-code or requirement-to-test), inter-requirement traceability is more textual and therefore particularly amenable to NLP methods. Moreover, by focusing on hierarchical refinement relations, we align with both the taxonomic semantics outlined by Ramesh and Jarke and the automation agenda championed by Guo et al. Our tool can thus be positioned as a concrete instantiation of the trajectory they describe: taking the conceptual insight that parent–child relations are semantically rich and practically valuable, and operationalizing it through NLP to reduce the capture burden in industrial settings.

Nevertheless, Guo et al. also caution against over-reliance on automation. They argue that NLP should not be seen as a replacement for human judgment but as an assistant that surfaces candidate links to be validated by engineers. This human-in-the-loop framing is particularly aligned with our design philosophy. By embedding the functionality in a lightweight extension, we do not aim to replace engineers' expertise but to augment it, offering a list of likely parent requirements that can be quickly reviewed. This approach addresses two of the persis-

tent barriers identified across the literature: it reduces the manual search effort (tackling the cost side of the cost–benefit equation) while preserving human oversight (thereby mitigating concerns about trust and accuracy).

In sum, the contribution of Guo et al. (2025) can be read as both a survey of techniques and a critical reflection on the state of the field. Its main strength is the articulation of how NLP has evolved from rudimentary keyword matching to sophisticated semantic modeling, and how these advances map onto different kinds of traceability tasks. Its main limitation is not internal but systemic: despite technical maturity, industrial uptake remains low, due to domain adaptation challenges, evaluation gaps, and the difficulty of integrating NLP into existing toolchains. For our research, this duality is invaluable. On the one hand, the chapter validates our focus on NLP-assisted father retrieval as timely and technically feasible. On the other hand, it warns us of the pitfalls of over-automation and the need to design for adoption rather than elegance. By acknowledging both sides, our work can position itself as a realistic step forward: not a revolutionary reimagining of traceability, but a pragmatic intervention that leverages NLP to make one of its most neglected aspects — inter-requirement hierarchical traceability — more accessible and more usable.

The literature reviewed in this chapter highlights the persistence and complexity of the requirements traceability problem. Early work by Gotel and Finkelstein underscored the lack of systematic support for pre-requirements specification traceability, particularly in capturing the motivations and rationales that underpin downstream requirements. Subsequent research, such as Ramesh's reference models, attempted to classify and formalize different traceability relations, but largely remained at a conceptual level, offering taxonomies rather than actionable mechanisms. Later efforts, including Spanoudakis and Zisman's rule-based approaches, made progress toward automation but remained brittle, relying on handcrafted syntactic patterns that were difficult to generalize across domains.

The advent of information retrieval (IR) and natural language processing (NLP) techniques brought renewed momentum, as shown in Guo et al.'s survey of NLP for traceability. While classical IR approaches based on TF–IDF and vector-space models provided a scalable foundation, they struggled with semantic precision, often linking requirements by superficial lexical overlap. Rule-based NLP systems improved semantic depth but introduced rigidity. More recently, embedding-based models leveraging neural language representations (e.g., BERT, Sentence-BERT) have demonstrated the ability to capture contextual meaning across hetero-

geneous requirement statements, enabling higher-quality retrieval and ranking.

Despite this evolution, significant gaps remain. Existing tools tend to focus on post-specification traceability (linking requirements to design, code, or test artifacts) rather than inter-requirement traceability that can clarify hierarchies and rationale. Moreover, industrial adoption is hampered by the integration burden of full-fledged requirements management suites and the lack of lightweight, task-specific tools that can be deployed alongside existing platforms.

This thesis builds upon these streams of work while diverging in two important ways. First, it focuses explicitly on pre-requirement interdependencies, particularly the retrieval of "father" requirements that express the motivation of more detailed "child" requirements. This positions the research squarely in the underexplored domain identified by Gotel and corroborated by subsequent scholars. Second, it operationalizes modern neural IR approaches within a lightweight external tool, capable of integrating with industrial suites such as Polarion through database import/export. By leveraging Hugging Face's embedding-based models for semantic retrieval, the tool exemplifies the latest generation of IR/NLP methods, but adapts them to the specific challenge of supporting requirement engineers in hierarchy building.

In doing so, this work builds directly on the foundations of IR-based traceability while addressing gaps left open by both rule-based approaches and traditional suites. It proposes not a wholesale replacement of existing requirements management systems, but a complementary capability that reduces the manual effort and cognitive load associated with establishing requirement hierarchies.

2.1 Existing Tools

Beyond the academic domain, several commercial tools have attempted to operationalize requirements traceability through integrated software solutions. Among the most prominent examples are The REUSE Company's RQA suite and Visure Solutions' Requirements ALM platform, both of which embed automated traceability mechanisms into broader requirement engineering environments. Despite their advanced user interfaces and integration with major lifecycle management tools, a closer inspection reveals that their underlying approaches remain largely rule-based and configurative, rather than fully semantic or learning-driven.

The REUSE Company provides one of the most sophisticated commercial implementations of semi-automated traceability. Within its ecosystem, trace links can be automatically

suggested across pairs of artifacts—typically between requirements, design elements, or test cases—based on linguistic similarity. This similarity is computed through what the company describes as an "NLP configuration for retrieval." In practice, this configuration defines how textual artifacts are preprocessed before comparison. It involves a series of classical natural language processing steps such as tokenization, lemmatization, part-of-speech tagging, stop-word filtering, and synonym expansion using configurable domain ontologies. Each preprocessing layer can be adjusted to emphasize certain grammatical or lexical features, and users can fine-tune similarity thresholds (for instance, displaying all links above 75% similarity). The system then suggests potential trace links based on computed similarity scores, which engineers manually validate or reject.

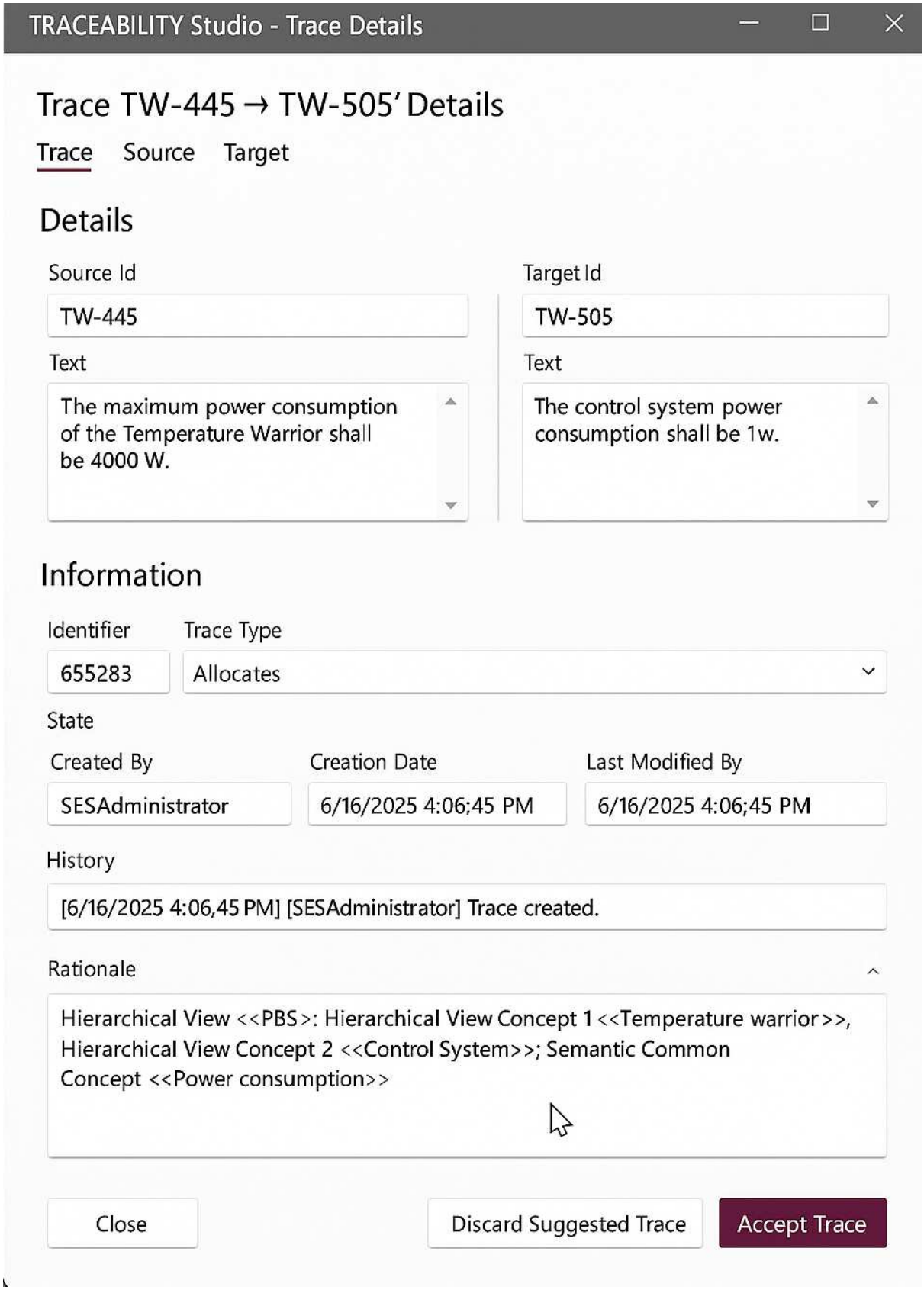

Figure 2.2: Suggested Trace

While this pipeline is linguistically rich, it remains deterministic and rule-driven rather than

adaptive. The REUSE Company's NLP configuration does not employ neural embeddings or transformer-based models such as BERT or RoBERTa. Instead, it relies on surface-level text matching and ontology-based reasoning to capture relationships between terms. The system recognizes equivalences such as "Car" $\approx$ "Vehicle" only when these have been explicitly defined in its lexicon and cannot infer deeper contextual or hierarchical relations between requirements that use different words or domain expressions. Thus, although the company's marketing refers to "NLP-based traceability," its underlying mechanism corresponds to a configurable pattern-matching framework, not a learned semantic model, based on proprietary algorithms or custom ones created by the user.

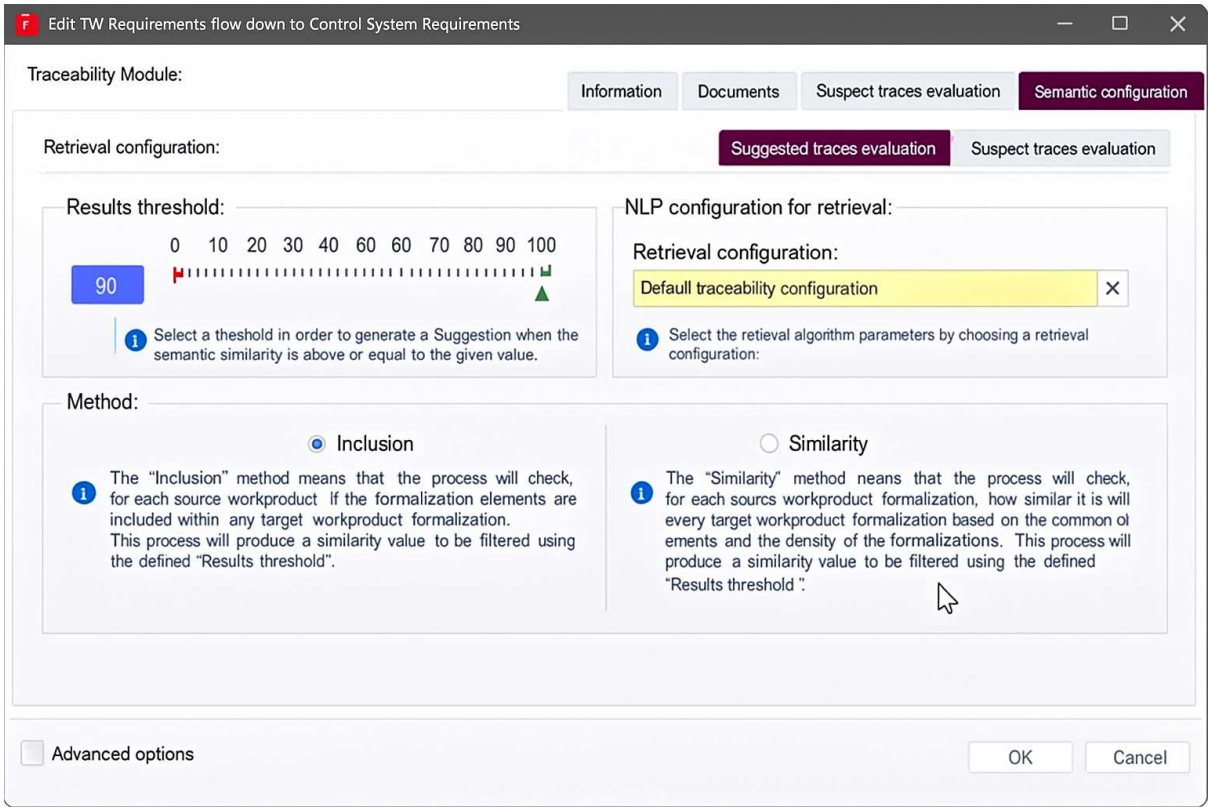

Figure 2.3: Semantic Module Dashboard

The process still requires user intervention both in defining the linguistic rules and in validating trace suggestions, resulting in a semi-automated, human-in-the-loop workflow.

A comparable philosophy underpins Visure Solutions' Requirements ALM platform, which integrates traceability within a complete lifecycle management environment. Visure focuses primarily on structural and impact-based traceability, enabling users to define relationships among requirements, risks, test cases, and design items through configurable templates. The system can automatically propagate changes, highlight inconsistencies, and generate traceabil-

ity matrices. However, Visure's automation is mostly relational rather than linguistic: trace links are inferred based on model relationships, attribute similarities, and metadata fields rather than the semantic content of the text. Although Visure has progressively introduced "AI-driven" modules for requirements quality and change prediction, their traceability engine remains predominantly dependent on predefined relationships and user-specified mappings rather than deep language understanding.

Taken together, both tools embody the current industrial frontier of semi-automated traceability: they leverage configurable algorithms and symbolic reasoning to reduce manual effort, yet they stop short of achieving true semantic inference. In contrast, the approach adopted in this thesis—based on transformer models from Hugging Face—moves beyond rule-based similarity toward contextual embedding-based semantic comparison. By encoding meaning directly in vector space rather than through preconfigured lexical rules, the proposed tool can autonomously identify conceptual and hierarchical relations, even when requirements are phrased heterogeneously or distributed across multiple documents. Moreover, whereas existing tools typically operate at the document-to-document level, the embedding-based method can scale to entire repositories, supporting hierarchical father–child detection that current industrial solutions only approximate. In this sense, while The REUSE Company and Visure Solutions represent advanced commercial implementations of traditional traceability paradigms, the present work positions itself as a step toward fully semantic, learning-based traceability that bridges the gap between linguistic similarity and genuine requirement understanding.

Chapter 3

Research Methodology

This chapter presents the methodological framework adopted to design, develop, and evaluate the tool that supports the identification of father–child relationships between requirements. The study follows a mixed-methods approach, integrating both quantitative and qualitative research components. This dual perspective is fundamental to properly assess a system that is simultaneously technical and human-centered: the tool must not only achieve measurable precision in identifying semantic links between requirements, but also prove its usefulness and usability for professional requirement engineers operating in real-world contexts. The chapter is organized as follows. The first part discusses the overall research design and its rationale. It then illustrates how the tool was conceived and developed, from the data collection and preprocessing stages to the underlying NLP model used for semantic similarity computation. The subsequent sections detail the experimental setup, including participant selection, validation procedures and key performance indicators. Finally, the chapter outlines how the data were analyzed and how validity and reliability were ensured throughout the process.

3.1 Research Design and Rationale

The study adopts a mixed-methods within-subjects design, where both objective performance metrics and subjective perceptions are assessed through complementary methodologies. The quantitative dimension focuses on evaluating measurable indicators such as time saved in requirement analysis, precision and recall in identifying correct father requirements. The qualitative dimension instead captures human-centered insights — how engineers perceive the tool's utility, usability, and integration potential within their workflow. This dual approach is justified

by the nature of the problem itself. Requirements traceability, especially at the pre-specification level, is not a purely computational problem; it inherently involves cognitive and interpretive reasoning by engineers. As highlighted in the literature, particularly by Gotel and Ramesh, one of the major barriers in effective traceability lies in the absence of tools that support the reasoning process behind requirements rather than merely tracking document-level relationships. The proposed tool aims to fill this gap, acting as an intelligent assistant that suggests likely "father" requirements — those that capture the rationale or higher-level motivation behind a newly written one. Therefore, both the technical accuracy of the retrieval mechanism and the perceived cognitive support it provides must be examined.

3.2 Tool Development and Architecture

The tool developed for this study was built on top of the Hugging Face NLP ecosystem, leveraging transformer-based embeddings for semantic similarity. Its goal is to support requirement engineers in the identification of motivational or hierarchical links — the so-called "father" relationships — by analyzing the semantic proximity of textual descriptions in a repository of requirements.

The development process began with the export of requirements from Polarion, the industrial requirements management system used in the project. These requirements were stored in a structured format (CSV) including identifiers, text, authorship metadata, and other attributes. The preprocessing phase was deliberately minimal, in line with best practices for transformer-based models, so as to preserve the domain-specific terminology used in the automotive and heavy-vehicle industry. Text normalization included only whitespace correction and lowercasing, avoiding lemmatization or stop-word removal that could disrupt the technical content.

Each requirement text was then converted into a dense numerical vector using a pre-trained sentence embedding model available on Hugging Face, typically a Sentence-BERT or MiniLM variant. These embeddings capture contextual meaning rather than mere lexical overlap, enabling the comparison of semantically related requirements even when they share few surface words. All embeddings were stored in an approximate nearest neighbor (ANN) index, such as Faiss, to allow rapid similarity searches across large repositories.

A graphical user interface (GUI) was coded in Phyton to make the tool as friendly as possible and lower the steepness of the learning curve of its utilization.

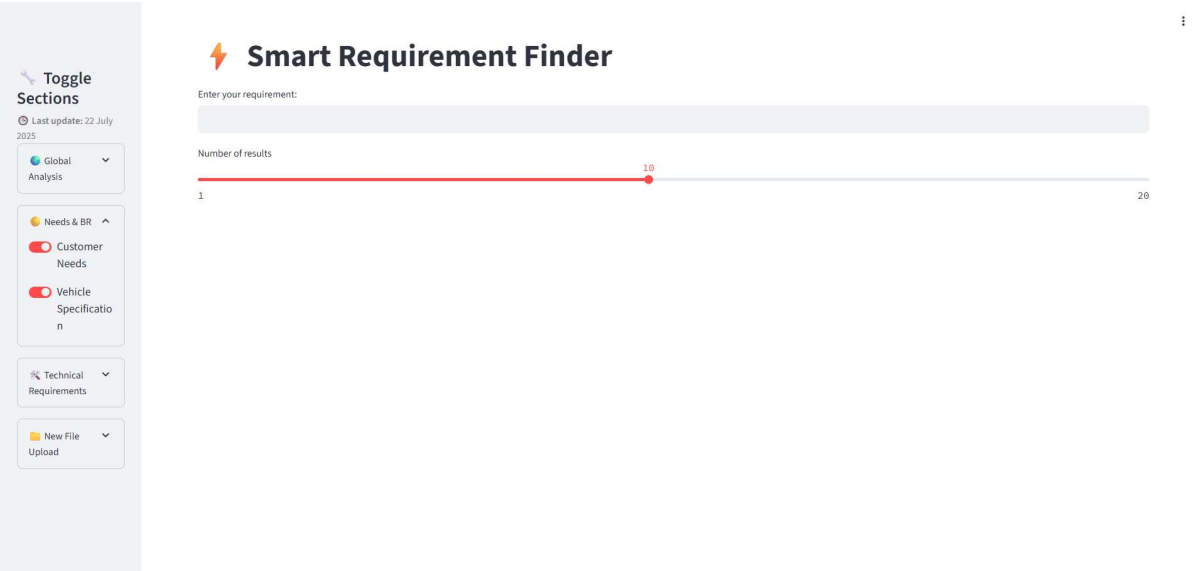

Figure 3.1: Tool's GUI

In the picture above, the User Interface is shown. On the left there are toggle settings that allow the user to choose where the tool shall look for suggestions, either in the global database, or in single documents. Below the query bar, in addition, a slider allows the choice of the number of suggestions to be made by the tool.

When a user enters a new requirement, the tool encodes it using the same model and queries the index to retrieve the most semantically similar existing requirements. The system returns the top results ranked by cosine similarity. The engineer can then review these suggestions, identify whether any of them represents a true father requirement. This workflow maintains human oversight while accelerating the initial exploration phase, which usually takes substantial manual effort.

The underlying logic is not limited to lexical matching, as in traditional traceability systems, but relies on deep semantic representations. This allows the tool to capture paraphrasing, conceptual similarity, or implicit motivational relations that might be overlooked by keyword-based approaches. The architecture can easily be extended with a cross-encoder re-ranking layer, which uses a more computationally expensive transformer to re-evaluate the top results for even higher precision. However, in the current prototype, the focus remains on the bi-encoder retrieval pipeline to ensure interactive performance.

3.3 Data and Ground Truth

The dataset used for evaluation was derived from a real industrial repository of requirements from a heavy-truck automotive project to create a reliable ground truth for validation.

3.4 Participants and Experimental Procedure

The experimental validation involved a team of six professional requirement engineers working within the same industrial environment. All participants possessed prior experience in systems requirements engineering and were familiar with the use of Polarion as their main requirements management platform. This ensured that the evaluation reflected real-world conditions rather than an artificial lab scenario. The engineers were divided into two groups of three:

- one group performed the task using the proposed NLP-based tool;
- the other group worked manually, relying only on Polarion's built-in search and navigation functionalities.

This between-groups configuration was chosen to prevent learning or memory effects that might have arisen if the same participants repeated the same tasks under both conditions. Each participant was given a set of requirements with already known father links, previously validated by domain experts and used as a reference ground truth. The goal of the experiment was to determine how effectively and efficiently each engineer could identify these father requirements within a fixed time frame of one hour. This setup allowed a direct comparison of performance between the two groups in terms of productivity (number of total requirements processed). For the group using the tool, the engineers could access the interface that displayed semantic similarity suggestions based on the embedding-based retrieval mechanism. For each "child" requirement, the system automatically proposed a ranked list of potential "father" requirements. The engineers were asked to evaluate these suggestions, select the most appropriate father where applicable. The control group, on the other hand, had to perform the same task manually — by searching through the repository, using keyword queries or browsing hierarchies, as they would normally do in their daily workflow. Both groups operated under identical conditions regarding data availability, time limit, and task difficulty. At the end of the session, quantitative results were collected on the total number of processed requirements. This metric enabled a comparison of performance across groups to measure efficiency (speed

of completion). After completing the task, participants using the tool were invited to fill out a short feedback questionnaire assessing their subjective experience. The engineers provided qualitative feedback on usability, perceived support in their reasoning process. These insights complemented the quantitative results, providing a richer understanding of the tool's potential impact on real-world engineering workflows.

3.5 Evaluation Metrics and Analysis

To rigorously assess the performance and reliability of the developed tool, two complementary evaluation perspectives were adopted: a quantitative assessment based on key performance indicators (KPIs), and a qualitative evaluation derived from user feedback and observation. The quantitative assessment was carried out using two distinct datasets and two levels of analysis. The first level — involving the team of six requirement engineers — was designed to compare the efficiency and usability of the tool against the standard manual approach. The second level — performed directly by the researcher — focused on a deeper analysis of retrieval quality, using well-established information retrieval metrics such as precision, recall, and success rate. In the team experiment, the comparison between the two groups (manual vs. tool-assisted) aimed to evaluate time saving and productivity. The key metric was the average number of correct father requirements identified within one hour, supported by the ratio of correct links over total links proposed. This measure provided an indication of how much the tool accelerated the process without compromising the reliability of the results. For the quantitative analysis performed by the researcher, a controlled dataset was used where the correct father–child relationships had already been validated. The output of the tool — consisting of ranked similarity scores between each requirement and all others in the repository — was compared against this reference. From this, three KPIs were derived:

- **Precision (P)**: the proportion of retrieved links that were correct, i.e., how many of the system's suggestions matched the ground truth.

$$\text{Precision} = \frac{\text{Number of correct fathers}}{\text{Total fathers found by the tool}}$$

- **Recall (R)**: the proportion of all existing correct links that were successfully retrieved by

the system.

$$\text{Recall} = \frac{\text{Number of correct fathers}}{\text{Total actual fathers}}$$

- **Success Rate (SR)**: a higher-level indicator defined as the fraction of child requirements for which the tool correctly identified at least one valid father link among the top suggestions.

$$\text{Success Rate} = \frac{\text{Number of queries with at least one correct father}}{\text{Total number of queries}}$$

Together, these metrics allowed a nuanced assessment of both the accuracy and the coverage of the tool's retrieval mechanism. Precision captured the reliability of its recommendations; recall assessed its ability to detect all relevant relations; and success rate offered an interpretable measure of the tool's practical usefulness in day-to-day requirement analysis. Finally, to contextualize these quantitative findings, the engineers' qualitative feedback was also analyzed. Their responses helped interpret the metrics in light of real human experience — for example, whether slightly lower precision might still be acceptable if the tool significantly reduced search time, or whether high recall led to information overload. This combination of objective and subjective evidence was essential to assess the actual value of the tool within a realistic engineering workflow.

Chapter 4

Results

4.1 Comparative Evaluation

To assess the effectiveness of the developed tool in supporting the retrieval of hierarchical ("father–child") relationships among requirements, a controlled empirical evaluation was conducted involving a group of six requirement engineers. The purpose of this experiment was to determine whether the integration of the NLP-based retrieval system could provide a measurable improvement in the efficiency of traceability analysis, compared to the traditional manual approach currently employed within the organization.

The experiment was designed as a comparative test between two homogeneous groups: three engineers using the tool (the experimental group) and three engineers performing the same task manually (the control group). The participants were all professionals with comparable experience levels in requirements analysis and familiar with the company's documentation practices. Each engineer was assigned the same set of requirements, selected from an industrial dataset where the correct "father" relationships had already been validated by domain experts. This provided a consistent ground truth for evaluation, ensuring that performance differences could be attributed to the method used rather than variations in data or expertise.

Each participant was given a fixed time constraint of one hour, during which they were asked to identify as many father–child links as possible. The engineers in the control group were allowed to rely only on conventional search techniques, such as reading through documents, scanning for key terms, or cross-referencing requirement identifiers in Polarion exports. Conversely, the engineers in the experimental group were given access to the developed NLP-based tool, which uses semantic similarity to suggest potential father requirements for any se-

lected child requirement. The tool displayed, for each query, a ranked list of candidate parents, allowing users to quickly inspect and validate the most plausible matches.

The setup intentionally mirrored a realistic working scenario, where engineers must analyze a large repository of requirements under time constraints. The main objective at this stage was not to assess the quality of the tool's suggestions—this would later be measured through precision and recall—but rather to evaluate its impact on productivity, expressed as the number of queries successfully resolved within the given time. In this sense, the test aimed to capture how much the tool could accelerate the process of locating hierarchical links between requirements.

At the end of the one-hour session, the results were consolidated and compared. The manual group, relying exclusively on traditional search techniques, collectively identified **47** father–child relationships across the dataset. This corresponds to an average of approximately **15.7** links per engineer per hour. In contrast, the tool-assisted group achieved a total of **59** links in the same time frame, or an average of **19.7** links per engineer per hour.

The difference between the two groups corresponds to an increase of roughly **25.5%** in productivity when using the NLP-based tool.

ID	TOOL	Solved Queries
Eng. 1	N	15
Eng. 2	N	20
Eng. 3	N	12
Eng. 4	Y	20
Eng. 5	Y	23
Eng. 6	Y	16

Total Manual	47	Req/h
Total TOOL	59	Req/h
AvgReq Manual per Engineer	15.67	Req/h
AvgReq TOOL per Engineer	19.67	Req/h
% increase in productivity	25.5%	

Figure 4.1: Test Results

This improvement is particularly meaningful given that all participants were operating un-

der identical conditions and that the total time available was strictly limited. Even within this short timeframe, the tool demonstrated a clear capacity to enhance the rate at which engineers could identify father–child associations.

Several qualitative observations emerged during the debriefing sessions. Participants in the experimental group highlighted that the tool was particularly helpful in the initial phase of exploration, where identifying potential father requirements often requires sifting through large volumes of text. By presenting a ranked list of candidate parents, the tool effectively reduced the cognitive load associated with this step, providing a more focused starting point for deeper analysis. One engineer commented that "the tool saves time by putting the likely fathers right in front of you—you still have to check them, but you're no longer starting from zero."

In contrast, engineers in the manual group described their process as "tedious" and "repetitive," often involving re-reading similar requirement sections and verifying potential overlaps manually. They also noted the difficulty of recognizing semantically related requirements that did not share exact wording—an area where the NLP-based system proved most advantageous.

Interestingly, during post-test interviews, even the engineers in the control group acknowledged that semantic retrieval could substantially improve traceability workflows. Several participants noted that while the manual approach forced them to rely on memory and keyword scanning, the tool could detect latent similarities that would otherwise be overlooked. This was particularly evident when requirements expressed equivalent concepts using different terminology or phrasing.

The analysis of individual performance further reinforced these observations. Although some variation existed among participants, **almost every engineer** in the tool-assisted group **outperformed** their manual counterpart in the number of links identified. This consistency suggests that the improvement was not due to individual skill differences but to the inherent advantage provided by the tool. Moreover, the effect appeared to scale with the engineers' familiarity with the interface: after only a short familiarization period, participants reported being able to operate the system intuitively, relying primarily on the similarity rankings to prioritize their review process.

From an organizational perspective, these findings provide strong evidence that the adoption of NLP-based retrieval in the traceability phase could lead to measurable efficiency gains. Even if the observed 25% productivity improvement were maintained across a large-scale requirements repository, the potential time savings during verification and analysis phases would

be substantial. In industrial environments such as automotive system development—where a single project can involve tens of thousands of requirements—the cumulative effect of such improvements could translate into a significant reduction in project overhead and human resource allocation.

In summary, the comparative experiment with the six engineers demonstrated that the tool provides a tangible advantage in terms of efficiency, enabling users to identify a higher number of correct father–child relationships in less time. The results confirm that NLP-based semantic retrieval can meaningfully augment the requirements engineer's workflow, particularly during early traceability exploration. This experiment thus serves as the first empirical validation of the tool's potential, while the subsequent analyses—focusing on precision, recall, and user satisfaction—provide deeper insights into its accuracy, reliability, and usability

4.2 Additional KPIs: Precision, Recall and Success Rate

To complement the engineer-led comparison described above, the tool was evaluated in a repository-level, offline experiment that measures classical information-retrieval metrics: **precision**, **recall**, and an application-oriented **success rate**. The rationale for this second experiment is straightforward: while the one-hour human trial captured differences in productivity under time pressure, it does not by itself quantify how faithfully the tool reproduces the ground-truth father relationships that already exist in the project's authoritative Polarion repository. The repository-level evaluation therefore asks a different question — given an approved corpus of requirements with validated father links, how often does the tool recover those links among its top-N suggestions? — and it is designed to produce reproducible, numeric KPIs that can be reported and compared across retrieval parameter settings.

4.2.1 Dataset and Setup

The input data for this evaluation are the subset of requirements exported from Polarion that have already been validated and approved in the project lifecycle. Each requirement record in the exported dataset contains the requirement identifier, the full textual description (title + body where available), and the Polarion-assigned father links (zero, one, or multiple). Prior to running the experiments, the dataset was cleaned and normalized to remove incidental formatting differences while preserving domain terminology: whitespace normalization, removal

of non-informative markup, and anonymization of any sensitive identifiers were applied. The complete preprocessing pipeline, the exact export snapshot (date and Polarion revision), and the identifiers of the records used in the experiment are archived so that the evaluation can be reproduced.

For each experiment run the process is the same. A requirement **q** is selected as the query (the "son"); the tool encodes **q** into its semantic vector representation and retrieves the top-N most similar requirements from the repository, ordered by cosine similarity. We repeat this for every query requirement in the evaluated set. The parameter ***N*** is varied across the following values: 5, 10, 15, and 20. These values were chosen because they strike a practical balance between engineer attention (how many candidates a user can realistically inspect) and the ability of the system to surface the relevant fathers. For each N the retrievals are recorded in an Excel sheet that mirrors the structure of the exported repository and contains explicit columns for the subsequent KPI calculations.

4.2.2 Data recording and Excel scheme

The experimental outputs are organized in a single spreadsheet per ***N*** value; each spreadsheet has one row per query requirement and the following columns (this exact schema is kept consistent across all runs):

1. **Number of Queries *(K)***: counter of the queries done through the tool;
2. **Requirement ID**: unique identification number assigned to each requirement;
3. **N**: number of requirements to be suggested by the tool, with

$$N \in \{5, 10, 15, 20\}$$

4. **Number of True Fathers**: number of father requirements found inside Polarion;
5. **Number of Fathers Found**: number of father requirements found with the tool assistance;
6. **Success**: 0 if no father requirements found, 1 if at least one father was identified;
7. **Precision**: fraction of the number of fathers found with the tool and N;

8. **Recall**: fraction of the number of fathers found with the tool and the number of true fathers.

Queries (K)	ID	N	Number of True Fathers	Number of Fathers Found	SUCCESS (1/0)	PRECISION	RECALL
1		5					
		10					
		15					
		20					
2		5					
		10					
		15					
		20					

Figure 4.2: Excel Table

An additional column then tracks the average scores obtained with respect to the changing in ***N***, to allow for further comments on the quality of the tool.

4.2.3 Metric decision and averaging policy

For each query K and relative ***N*** the per-query measures *Precision at N*, *Recall at N*, *Success at N* as described above; the global KPIs are calculated as follows:

- **Macro-Averaged Precision @N**:

$$\text{Precision@N}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K}\text{Precision@N}_k$$

- **Macro-Averaged Recall @N**:

$$\text{Recall@N}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K}\text{Recall@N}_k$$

- **Macro-Averaged Success Rate @N**:

$$\text{SuccessRate@N}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K}\text{Success@N}_k$$

In this way, both per-query and macro KPIs are calculated. Results regarding the trade-offs of varying the number of requirements given by the tool can be highlighted. While the previous averaging policies were based on the parameter N, corresponding to the number

of retrieved candidates per query, a complementary analysis was performed to evaluate how the number of real father requirements affected the tool's performance. The objective of this additional aggregation was to investigate whether the hierarchical complexity of a requirement — measured by how many father requirements it actually possesses — influences the capability of the model to correctly retrieve meaningful links.

To perform this analysis, the complete dataset of evaluated requirements was divided into subsets according to their number of validated fathers, as defined in the Polarion database. For each subset, the same key performance indicators (precision, recall, and success rate) were computed using the same definitions introduced earlier. However, instead of averaging across N-values, the aggregation here was performed across all requirements sharing the same number of fathers, irrespective of the number of retrieved suggestions N.

This allowed the calculation of average precision, recall, and success rate per complexity level, providing a different lens through which to interpret the model's robustness. In particular, the goal was to observe whether the tool tends to perform better on simpler requirements — those with a single or few fathers — or whether its semantic similarity mechanism scales effectively when multiple parent relationships exist. The dataset was therefore segmented into groups (e.g., one, two, three, or more real fathers), and each group's KPIs were computed by averaging the individual scores obtained for the requirements in that group, and then computing the global average.

By structuring the analysis in this way, the study isolates the influence of structural complexity from the influence of N, enabling a clearer understanding of where the tool provides the most tangible benefit. The results of this investigation — expressed as average precision, recall, and success rate for each group — are presented and discussed in the following section.

4.2.4 Experimental procedure

For the gathering of the results, the tool was utilized over a set of 50 requirements, varying the number *N*. The column *"Number of True Fathers"* was filled with the number of fathers that each requirements was assigned in Polarion, usually ranging between 1 and 3. The requirement titles were then given to the tool as input: among the suggestions, the true fathers were looked for, in order to compile the *"Number of Fathers found*. All the other columns are then automatically filled, after setting Excel formulas for the whole columns, in particular:

- Success (1/0): is automatically filled by the Excel formula:

=SE(Number of fathers found > 0, 1, 0)

- Precision: filled by dividing the Number of Fathers found by the N column;
- Recall: done dividing the *Number of fathers found* by the *Number of Real Fathers*.

As an example, in the Figure below:

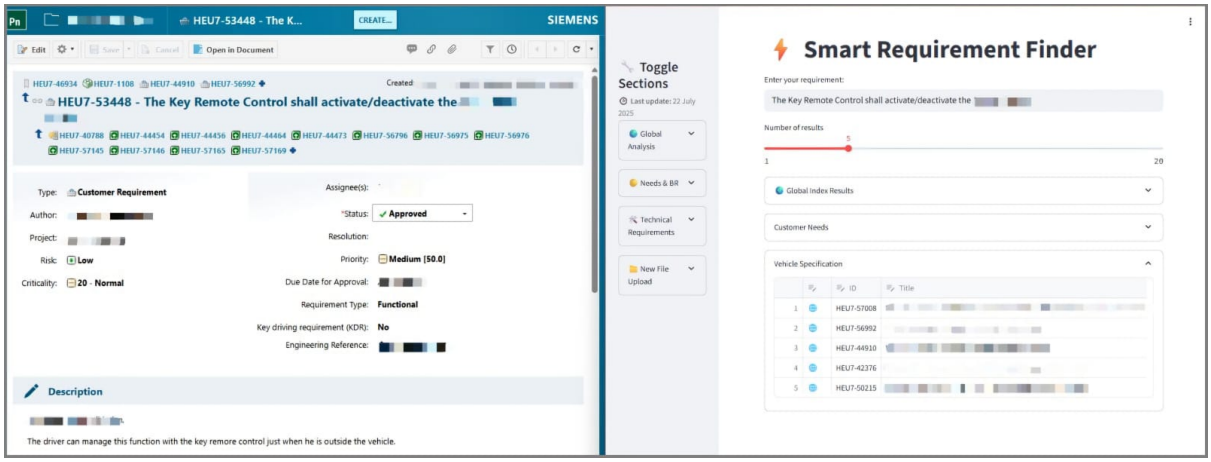

Figure 4.3: Polarion and Tool visualization

on the left we have the requirement taken in Polarion and on the right the same requirement inserted in the Tool. The father requirements in Polarion are represented above the title, with the IDs: ***HEU7-44910*** and ***HEU7-56992***. On the right we see them in the list proposed by the tool as second and third results. In this case the Excel sheet was filled in this way:

Queries (K)	ID	N	Number of True Fathers	Number of Fathers Found	SUCCESS (1/0)	PRECISION	RECALL
23	HEU7-53448	5	2	2	1	40.00%	100.00%
		10	2	2	1	20.00%	100.00%
		15	2	2	1	13.33%	100.00%
		20	2	2	1	10.00%	100.00%

Figure 4.4: Excel query results example

4.3 Aggregation and Results reporting

In this section, the actual results obtiained from the tool's KPI analysis are displayed.

N	Macro Precision	Macro Recall	Macro S.R.
5	12.80%	41.00%	54.00%
10	9.30%	59.83%	76.00%
15	7.20%	67.17%	82.00%
20	6.40%	79.50%	90.00%

Table 4.1: Macro KPIs calculated @N

This table was then exploited to plot a graph clearly and visually highlighting the increasing of the KPIs, varying the number N.

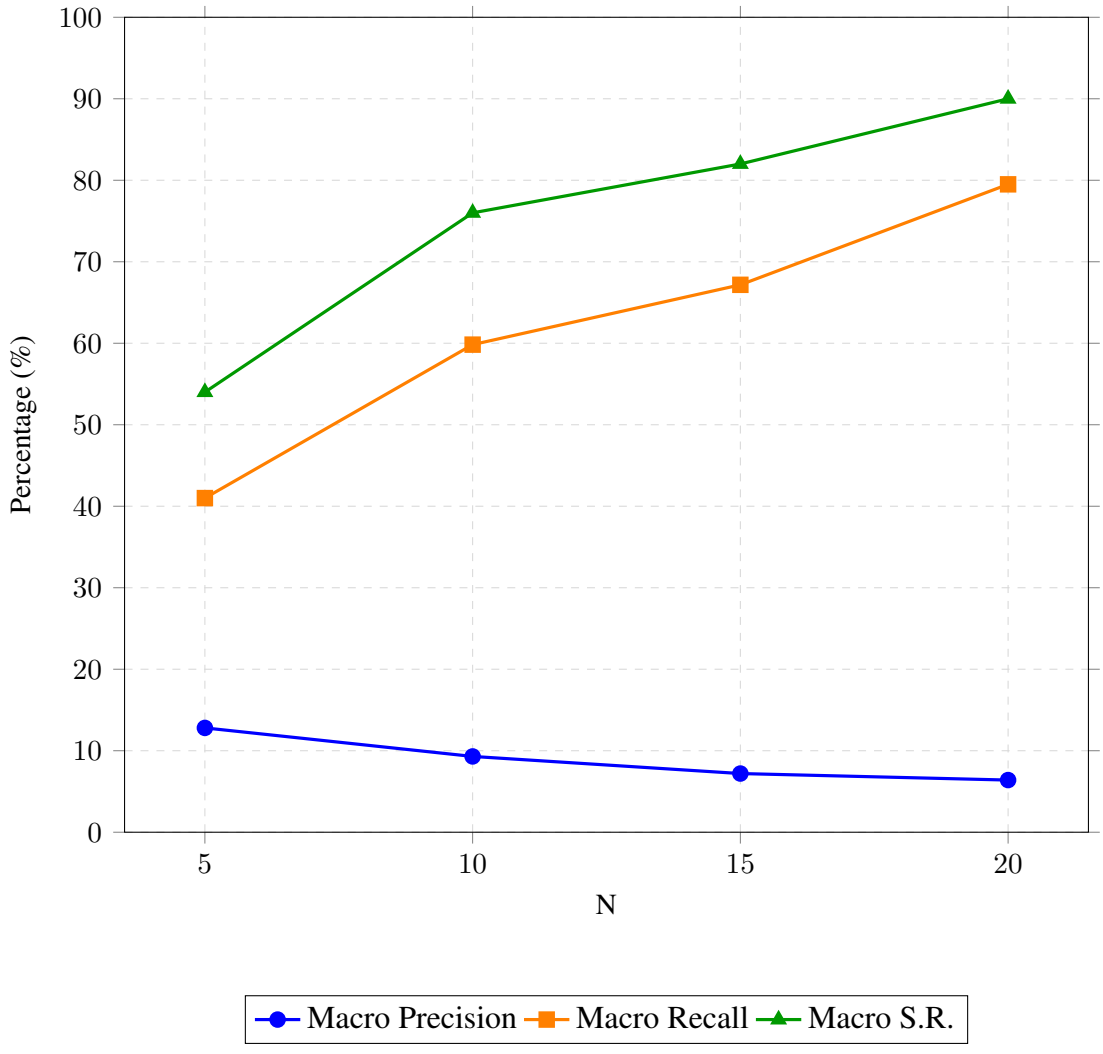

Figure 4.5: Comparison of Macro Precision, Recall, and Success Rate across N values.

Number of real fathers	Sample Number	Avg Precision	Avg Recall	Avg S.R.
1	23	6.39	69.57	69.57
2	20	10.67	58.13	76.25
3	6	13.61	50.00	95.83
4	1	7.92	31.25	75.00

Table 4.2: Macro KPIs calculated @ Number of real fathers

Chapter 5

Discussion

The discussion chapter aims to interpret and contextualize the results obtained throughout the experimental phase, linking them back to the central research objectives and the existing body of literature.

Whereas the previous sections focused on presenting empirical findings in an objective manner, this part delves into their broader meaning and implications — both theoretical and practical. The goal is to assess how the observed outcomes validate, refine, or challenge existing knowledge about the automation of requirements traceability, particularly within the pre-traceability and inter-requirement domains highlighted earlier in the state of the art.

Through a critical analysis of the measured KPIs — precision, recall, success rate, and time efficiency — the discussion seeks to uncover not only the strengths and limitations of the implemented tool, but also what these findings reveal about the current maturity and potential of NLP-driven approaches to hierarchical requirement analysis.

By doing so, the chapter bridges the gap between experimental evidence and its significance for both academic inquiry and industrial practice, offering an integrated understanding of how semantic retrieval technologies can enhance the everyday activities of requirement engineers.

5.1 Main and Secondary Results

The results obtained from the experimental phase offer a multifaceted understanding of the real effectiveness of the developed requirement-retrieval tool when compared to traditional manual methods. The evaluation was designed with the twofold objective of assessing, first, the efficiency gain introduced by the tool during father-requirement identification tasks, and sec-

ond, the accuracy and reliability of the retrieved links when compared to validated references already stored in Polarion. These two dimensions correspond respectively to the productivity improvement test conducted with a group of engineers and to the quantitative performance assessment using precision, recall, and success rate indicators.

The first set of results originates from the controlled experiment carried out with six requirement engineers, divided into two balanced groups of three participants each. The engineers were all familiar with the Polarion environment and shared a comparable level of expertise in requirement management activities. Both groups were given the same dataset of requirements containing a number of items whose father relations were already known but not explicitly disclosed to them. The goal was to identify as many father requirements as possible within a one-hour session. One group performed the task entirely manually, following the standard Polarion workflow of textual search, navigation through linked documents, and contextual reading. The second group carried out the same task with the assistance of the developed semantic-retrieval tool, which provided ranked suggestions for potential father requirements based on semantic similarity between requirement descriptions.

The results of this comparison revealed a clear difference in productivity. The group operating without the tool was able to correctly identify a total of 47 father relations, while the team supported by the tool reached 59 correct identifications within the same time frame, this corresponds to an average of 19.7 requirements per engineer in the assisted condition versus 15.7 in the manual condition, yielding a relative productivity improvement of roughly 25 %. This difference is particularly notable given the short duration of the task, which suggests that the learning curve associated with the tool is minimal. Engineers could adapt to its operation within minutes and benefit almost immediately from the suggestions provided by the system.

From a qualitative standpoint, several behavioral patterns were observed during the experiment. The engineers using the tool tended to rely on its ranked outputs as a starting point for deeper verification, spending less time navigating among documents and more time assessing whether the proposed relations were semantically and functionally correct. In contrast, those working manually had to perform more exploratory searches, often revisiting the same requirement sections multiple times. This shift in focus—from searching to validating—represents an important organizational improvement in the traceability process. It suggests that automated retrieval does not replace engineering judgment but instead redefines how it is applied: the tool reduces mechanical effort and allows human expertise to concentrate on reasoning and

validation.

Although the test was limited to a small number of participants, the observed improvement is consistent with the long-standing findings in the literature that manual traceability tasks are error-prone and time-consuming (Gotel and Finkelstein, 1994; Ramesh and Jarke, 2001). The quantitative results provide empirical confirmation that the integration of NLP-based similarity models within existing requirement repositories can concretely address the "pre-traceability" gap identified by Gotel and later reaffirmed by Guo et al. (2025). In this sense, the test demonstrates that semantic retrieval can operate as a feasible and effective first layer of automation in requirement-to-requirement linkage analysis. Beyond the productivity experiment, further quantitative evaluation was conducted through the analysis of three key performance indicators: precision, recall, and success rate. These metrics were computed by comparing the tool's suggested fathers with a dataset of validated father requirements extracted from Polarion. The procedure was designed to measure how accurately the tool can reproduce existing correct links and how its performance changes depending on the number of retrieved candidates (N). For each query requirement, the tool was asked to return the top N most semantically similar results, with N = 5, 10, 15, and 20. For each configuration, the retrieved fathers were compared against the real validated ones.

A detailed dataset was constructed in which each retrieved requirement was marked with a binary indicator (1 if at least one retrieved item corresponded to a true father in Polarion, 0 otherwise). From this, precision was defined as the ratio of correct fathers among the retrieved N items (number of true fathers / N), while recall was computed as the ratio of retrieved true fathers over the total number of true fathers existing in the repository for that requirement. For instance, if a requirement had three known fathers and the tool found two of them within its top-N results, recall would be $2/3 \approx 0.67$, while precision would be $2/10 = 0.2$. The success rate represented a higher-level indicator, expressing the proportion of requirements for which the tool successfully retrieved at least one correct father within the top N suggestions. Average values of precision, recall, and success rate were then calculated for each N configuration.

The results, summarized in the internal dataset, exhibit the expected **trade-off behavior between precision and recall**: as N increases, recall improves because more potential candidates are included, but precision tends to decrease due to the introduction of false positives. This pattern confirms that the retrieval model behaves in line with established information-retrieval principles. In the context of requirement engineering, however, the balance between precision

and recall assumes a specific significance: engineers often prefer higher recall, as missing a valid trace could mean losing a critical dependency, while false positives can be filtered out through quick validation. For this reason, the average recall and success rate becomes a particularly meaningful metric, indicating the probability that the tool can assist at least partially in the identification task. Across the tested configurations, success rate values consistently exceeded **90%**", confirming that even with simple semantic embeddings the system can reliably highlight relevant parent candidates.

A complementary layer of analysis can be derived by observing the aggregated KPIs in relation to the *number of validated father requirements* associated with each query, as summarized in Table "KPIs based on number of real fathers".

This table provides an alternative view of performance, not across retrieval depth (N), but across the inherent **structural complexity** of each requirement within the Polarion repository.

Such a perspective allows for a more nuanced understanding of how the model performs when confronted with different degrees of hierarchical dependency — from isolated, single-father cases to requirements connected to a broader web of parent elements. At a first glance, the absolute percentages of precision and recall in this table appear lower than those observed in the *per-N* analysis.

However, this difference is expected and primarily methodological: the KPIs were **averaged across all N values (5, 10, 15, 20)** for each group of requirements, effectively smoothing out peak values and reducing the overall magnitude of each metric.

This averaging strategy was adopted to ensure fairness and comparability, as it mitigates the bias that could emerge if only the best-performing N were considered.

In practice, it provides a more realistic estimate of the tool's baseline performance under variable retrieval configurations, reflecting what an engineer might observe in real usage without excessive fine-tuning.

The observed pattern across the four categories also aligns with expectations derived from the tool's semantic retrieval mechanism.

When the number of real fathers is **one**, recall and success rate hover around 69.6%, indicating that the model consistently identifies the correct parent relationship in most cases, though with modest precision (6.4%).

This again confirms the model's aptitude for simple one-to-one relational inference, where the semantic overlap between the child and its single father is strong and direct.

As the number of real fathers increases, however, recall and precision evolve asymmetrically.

For cases with **two or three fathers**, the average precision improves to around 10–13%, while recall gradually declines to approximately 50–58%, and success rate peaks at an impressive 95.8% for the three-father group.

This behavior suggests that in moderately complex structures, the model tends to **overgenerate relevant candidates**, successfully capturing at least one true relation per query, but not all of them.

In other words, as the semantic space becomes richer, the tool becomes better at "casting a wide net" but less precise in fully reconstructing all parent links.

The extremely high success rate in the three-father group, however, reinforces its *usefulness as a support system*: even when not exhaustive, the tool consistently identifies at least one valid father, providing valuable guidance for human reviewers.

The case with **four fathers** (albeit represented by a single data point) reflects a predictable saturation point, where recall (31.3%) and precision (7.9%) drop sharply.

This likely stems from both *semantic interference* — where shared phrasing between multiple parent requirements reduces discriminative clarity — and *statistical fragility* due to limited sample size.

It is worth noting that the Polarion dataset itself may introduce an element of approximation here: father-child relationships are defined at varying granularity levels, and in some cases, what the database lists as four distinct fathers might in fact represent **semantically redundant or overlapping entries**.

This structural noise makes the recall metric appear lower than it would under a perfectly curated hierarchy, as the model cannot distinguish between conceptual and administrative duplication.

Overall, Table "KPIs based on number of real fathers" confirms that the system behaves consistently across increasing relationship cardinalities and that performance degradation in more complex scenarios is largely attributable to the **intrinsic ambiguity and redundancy** of industrial requirement datasets, rather than to model instability.

By averaging results across N and grouping them by relational density, this analysis provides a realistic benchmark for the operational reliability of the tool in everyday use.

It highlights how, even under imperfect repository conditions, the model maintains a sta-

ble ability to identify meaningful semantic links — a crucial advantage in environments where engineers must frequently navigate incomplete or inconsistently structured requirement hierarchies.

In addition to the numerical analysis, a qualitative inspection of errors was conducted to better understand the tool's limitations. False positives were most frequently associated with requirements that shared domain terminology but belonged to different subsystems. For instance, terms like engine management and power management appeared semantically close but referred to distinct engineering functions. Such cases underline the challenge of purely textual similarity when technical vocabulary is reused across contexts. Conversely, some false negatives arose from very short requirement statements, where the lack of contextual information limited the semantic model's ability to establish meaningful similarity. These observations suggest that future iterations of the tool could benefit from context-aware embeddings or hierarchical fine-tuning on the company's internal corpus to better reflect domain-specific semantics.

Beyond the strictly quantitative outcomes, the experiment also produced a set of secondary results of both cognitive and organizational relevance. The post-experiment feedback gathered from participants indicates a generally positive perception of the tool's usefulness and usability. Engineers appreciated the intuitive nature of the interface and the organization of results in subjects, which allowed them to easily decide where to look for fathers. Several participants also remarked that the tool encouraged them to explore connections they might not have considered manually, suggesting a secondary benefit in stimulating broader system comprehension, as well as be utilized in the opposite way and help them find sons requirements that were not linked to a certain father. The main usability concern mentioned was the lack of filtering options to exclude already-linked or irrelevant requirements, a functionality that could be implemented in future development stages.

Another secondary observation relates to workflow integration. The import and export processes between Polarion and the external semantic-retrieval environment revealed a number of challenges concerning data formatting and metadata completeness. In particular, inconsistencies in the naming conventions of requirements or in the capitalization of attributes occasionally influenced the model's similarity computation. While these issues do not undermine the validity of the results, they point to the importance of standardizing requirement documentation formats to fully exploit AI-based retrieval tools. This also highlights one of the broader implications of the work: the success of semantic automation in requirement engineering is not

determined solely by algorithmic sophistication, but also by the quality and consistency of the textual data it processes.

Finally, from an organizational standpoint, the engineers' behavior during the tests provides insight into how automation reshapes the requirement analysis process. Rather than replacing the analyst's role, the tool re-allocates effort from exhaustive document search to critical evaluation. The task becomes less about finding and more about deciding. This shift reflects a fundamental change in cognitive engagement, aligning with modern systems-engineering principles that advocate for human–AI collaboration. In this sense, the observed improvements in speed are not merely quantitative gains but indicators of a deeper methodological transformation—one that enhances traceability without sacrificing human oversight.

In conclusion, the results collectively demonstrate that the implemented prototype meets its primary objective: to assist engineers in the identification of father requirements with measurable gains in efficiency and accuracy. The productivity test validated its impact in real operational contexts, while the precision–recall analysis confirmed its technical soundness against validated references. Together with the positive qualitative feedback, these findings substantiate the hypothesis that lightweight NLP-based tools can be effectively integrated into existing industrial workflows, bridging the pre-traceability gap and laying the foundation for more advanced semantic analysis in the future.

5.2 Contributions to the Theory

The analytical extension performed in this work — the construction and interpretation of a traceability matrix derived from a real industrial repository — represents not only a validation of the developed tool, but also a conceptual contribution to the theoretical understanding of requirement interdependencies and systemic propagation of change within large engineering projects. While the previous sections have focused on the quantitative performance of the NLP-based tool and its ability to retrieve hierarchical relations efficiently, this section shifts the attention to what these relations mean in theoretical terms. By visualizing and classifying the network of requirements, the study operationalizes the abstract idea of traceability as a socio-technical network, where each node (requirement) influences and is influenced by others through directional dependencies that reflect design intent, architectural hierarchy, and risk exposure.

5.2.1 Construction of the Matrix

The matrix was generated by exporting a subset of the requirement repository from Polarion, containing approximately 200 requirements belonging to a single system of the vehicle. Each requirement was identified by an `ID` and a list of `Linked Work Items`, where the nature of the relationship was explicitly stated through textual patterns such as *refines: ID"* and *is refined by: ID"*. These correspond, respectively, to the classical **downstream** and **upstream** directions of traceability: a "refines" link indicates that the current requirement is a child of another (i.e., more detailed or derived), whereas an "is refined by" link indicates a parent, or father, requirement.

Starting from these relations, a square adjacency matrix of size 200×200 was created, where both rows and columns represented the same set of requirements. The intersection cell (i,j) was filled with a symbol depending on the relation between requirement i and requirement j:

- The symbol ">" was used if requirement in the row i *refines* the one in column j, meaning i is a child of j;
- The symbol "<" was used if requirement in the row i *is refined by* the one in column j, meaning i is a father of j.

Through this formalization, the matrix acts as a topological map of the hierarchical relationships among requirements. The structure was then organized to approximate a *lower triangular* form, grouping related clusters so that higher-level requirements appear toward the lower region and progressively detailed requirements populate the upper-left. This visual arrangement is consistent with the concept of hierarchical decomposition in systems engineering, where abstraction decreases along the diagonal, it then facilitates the detection of hierarchical consistency: ideally, a well-structured system would show dependencies concentrated near the diagonal, reflecting coherent decomposition.

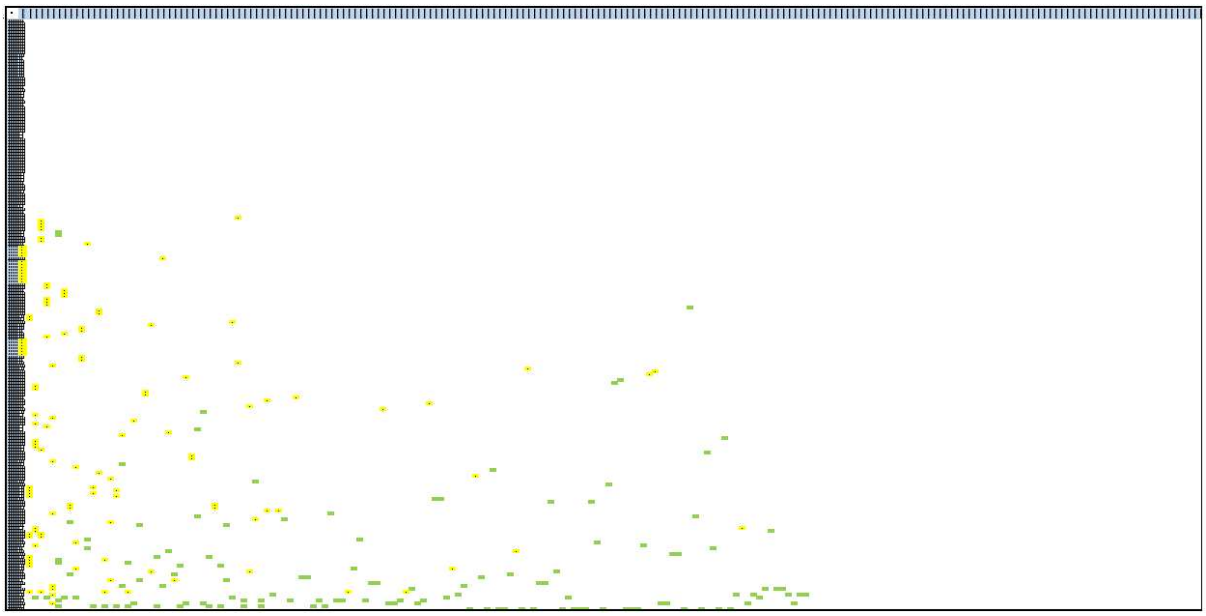

Figure 5.1: Adiacency Matrix

The cells containing the dependency symbols were color-coded to provide a clear visual overview of how the matrix is organized and to facilitate the identification of clusters or patterns among related requirements. In particular "<" relationship cells were assigned the color Green, while the ">" ones were filled in Yellow.

5.2.2 Quantitative Analysis and Classification

From the adjacency matrix, three primary numerical indicators were extracted for each requirement:

- **In-Degree**: Number of Fathers, counts how many requirements the given requirement depends on (incoming edges).
- **Out-Degree**: Number of Sons, counts how many requirements depend on this requirement (outgoing edges).
- **Total Degree**: counts how many dependencies each requirement has, both up and downstream.

These metrics capture the structural role of each requirement in the network. The "Number of Fathers" indicates how many higher-level elements a requirement depends on (its sensitivity),

while the o"Number of Sons" indicates how many lower-level elements depend on it (its influence). On the basis of these two indicators, each requirement was classified into one of four categories that represent distinct roles within the requirement network:

1. **Downstream-Critical**:requirements with high out-degree and low in-degree, i.e., those that refine multiple child requirements but depend on few fathers. These nodes act as drivers in the system: a change in them propagates widely downstream.

2. **Upstream-Sensitive**: requirements with low out-degree and high in-degree, typically representing elements that consolidate constraints from multiple higher-level items. They are receptors of upstream complexity and require careful synchronization.

3. **Equal-Impact**: requirements with roughly balanced in-degree and out-degree, situated in the middle layers of the hierarchy. They behave as mediators between different abstraction levels.

4. **Independent**: requirements with no explicit links, i.e., total degree equal to zero. These nodes are isolated in the network and might correspond either to stand-alone functionalities, to cross-vehicle requirements or to documentation gaps where traceability links are missing.

This categorical framework was then summarized in a pie chart, showing the proportion of requirements belonging to each class. The distribution itself offers a first-order picture of the system's traceability maturity: an ideal decomposition would show a healthy balance between the categories, while an excess of "independent" items might indicate poor documentation or disconnected requirement chains.

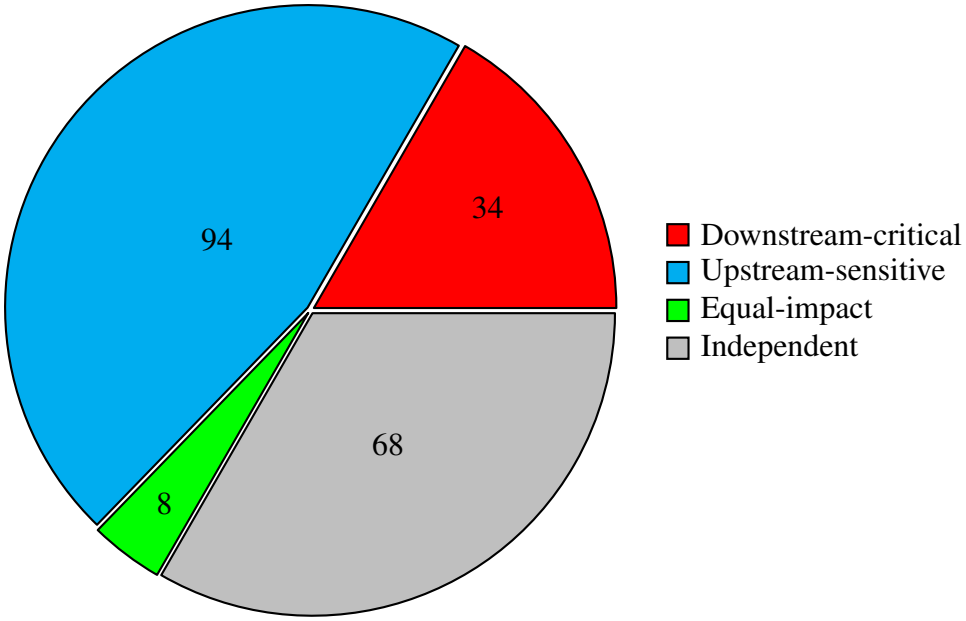

Figure 5.2: Distribution of requirements by category.

5.2.3 Network Construction and Analysis in Python

To extend the analytical depth of the traceability matrix and uncover systemic properties of the requirement repository, the dataset was imported into a Python-based environment and analyzed through computational graph theory. This step aimed to move beyond descriptive counting of upstream and downstream dependencies, translating the matrix into a dynamic representation of the requirement system as a directed network. Such transformation enables the application of quantitative techniques drawn from Social Network Analysis (SNA), providing a rigorous foundation for interpreting traceability as an emergent structure rather than as a static documentation artifact.

Data preparation. The adjacency matrix—previously constructed in Microsoft Excel—was exported in *.xlsx* format and processed using the *pandas* library in Python. Each row and column corresponded to a requirement identified by its unique *ID*, while the cell content indicated the presence and direction of a dependency. Cells containing the symbol “>” denoted that the requirement in the row refines the one in the column (downstream link), whereas cells marked with “<” indicated the opposite (upstream link). Empty cells were treated as an absence of

direct dependency. The resulting dataset therefore encoded the complete topological map of refinement relations among the 204 requirements considered in this study.

Graph generation. Using the `NetworkX` library, a directed graph $G(V, E)$ was constructed, where the vertex set V represents individual requirements and the edge set E represents refinement links between them. Each ">" symbol generated a directed edge from the child to its parent requirement, while each "<" generated the inverse relation. This ensured a consistent orientation of edges, following the convention that information flows from more detailed requirements toward higher-level abstractions. Once instantiated, the graph provided a machine-readable structure on which topological and centrality metrics could be computed.

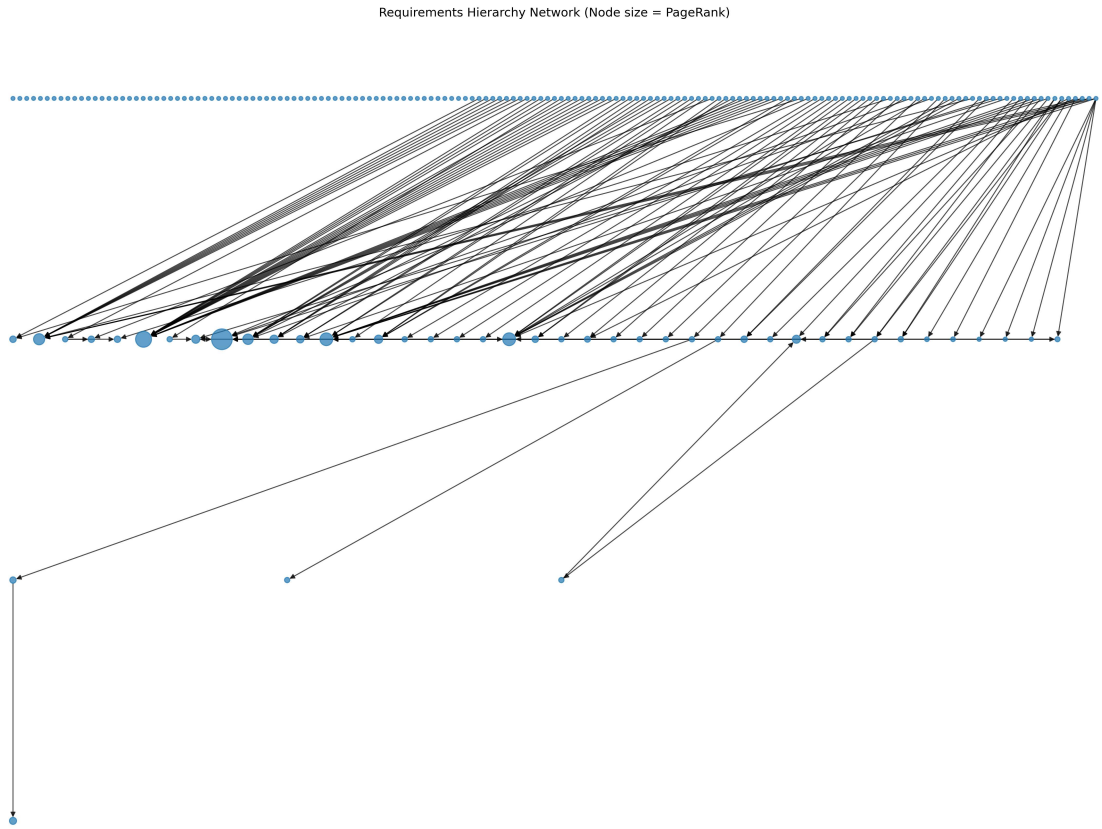

Figure 5.3: Directed Graph

Metric computation. The following structural indicators were extracted from the graph:

- **In-degree:** number of incoming edges, corresponding to the count of father requirements that refine the current node.

- **Out-degree:** number of outgoing edges, representing the number of child requirements that depend on the node.
- **Betweenness centrality:** quantifies the frequency with which a node lies on the shortest path between other nodes, identifying structural bridges or bottlenecks.
- **PageRank:** measures the relative influence of a requirement within the network by considering both the quantity and quality of its connections.
- **Hierarchy depth:** computed as the length of the longest directed path between a top-level node and a leaf node, representing the maximum number of refinement levels.

These metrics provide complementary perspectives: while in-degree and out-degree capture direct relational complexity, centrality measures highlight systemic influence and potential change propagation paths.

Computational workflow. All computations were performed within a Jupyter Notebook environment, enabling interactive data inspection, execution of Python code, and direct visualization of results. The process followed a structured sequence:

1. **Import of the dataset:** the Excel file was read using `pandas.read_excel()`, with requirement IDs set as both row and column indices.
2. **Graph creation:** for each cell containing ">" or "<", a directed edge was added to the graph according to the established directionality rules.
3. **Metric calculation:** functions such as *nx.in_degree()*, *nx.out_degree()*, *nx.betweenness_centrality()*, and *nx.pagerank()* were applied to obtain numerical indicators for every requirement.
4. **Depth assessment:** the longest path length was computed by iterating over all root nodes (requirements with no incoming edges) using *nx.single_source_shortest_path_length()*.
5. **Visualization:** using `matplotlib`, a hierarchical layout of the network was generated, with node size proportional to PageRank and arrows representing refinement direction.

Summary of results. The resulting directed graph comprised 204 nodes (requirements) and the complete set of refinement links. A series of structural metrics were computed to characterize the global properties of the network and the role of individual requirements. Table 5.1 summarizes the principal indicators obtained from the analysis.

Metric	**Value / Description**
Number of nodes (requirements)	204
Average in-degree	0.63
Average out-degree	0.63
Network density	0.0031
Top-level requirements (no incoming edges)	159
Leaf requirements (no outgoing edges)	95
Maximum refinement depth	3 levels
Most influential nodes (PageRank)	HEU7-18520 (0.0839), HEU7-40919 (0.0501), HEU7-18555 (0.0323)

Table 5.1: Summary of global network indicators.

The results indicate a sparse and weakly connected structure, where most requirements participate in few direct dependencies and only a limited number of nodes act as central anchors. The maximum refinement depth of three levels confirms that the specification remains shallow, suggesting a modular decomposition approach consistent with distributed system architectures. The relatively low density (0.0031) implies that refinement relationships are concentrated within local clusters rather than uniformly spread across the network.

To identify elements that serve as bridges between otherwise disconnected regions, *betweenness centrality* was computed for all nodes. Table 5.2 reports the ten requirements with the highest centrality values, representing potential bottlenecks for information or change propagation. Despite their structural significance, the absolute centrality values remain low (all below 3×10^{-4}), confirming that the network is only weakly centralized.

Rank	Requirement ID	Betweenness Centrality
1	HEU7-18555	2.9264×10^{-4}
2	HEU7-44078	1.7071×10^{-4}
3	HEU7-969	1.2193×10^{-4}
4	HEU7-44010	9.7547×10^{-5}
5	HEU7-55997	9.7547×10^{-5}
6	HEU7-44059	7.3160×10^{-5}
7	HEU7-44985	4.8773×10^{-5}
8	HEU7-42473	4.8773×10^{-5}
9	HEU7-39311	4.8773×10^{-5}
10	HEU7-27404	4.8773×10^{-5}

Table 5.2: Top 10 requirements ranked by betweenness centrality, computed in Python using the *NetworkX* library.

Together, these results portray a requirement network that is sparse, shallow, and loosely coupled. A small number of nodes serve as cross-linking elements, while the majority remain localized within independent refinement branches. This configuration is consistent with modular and resilient system architectures, where structural independence between clusters limits the propagation of change and reduces overall complexity. To complement the quantitative indicators, the resulting graph was visualized using *Matplotlib* and *NetworkX*, with node size proportional to PageRank centrality and arrow direction representing the refinement hierarchy. This graphical representation, shown in Figure 5.4, provides an intuitive confirmation of the numerical results. Clusters of closely related requirements emerge clearly, while only a few nodes appear as connectors between otherwise independent groups. The overall layout visually reinforces the interpretation of a sparse, weakly centralized network, consistent with the modular and hierarchical structure suggested by the previous metrics. Node size is proportional to PageRank centrality, while arrows represent the direction of refinement. The visualization highlights the modular structure and the limited number of highly connected nodes acting as traceability anchors.

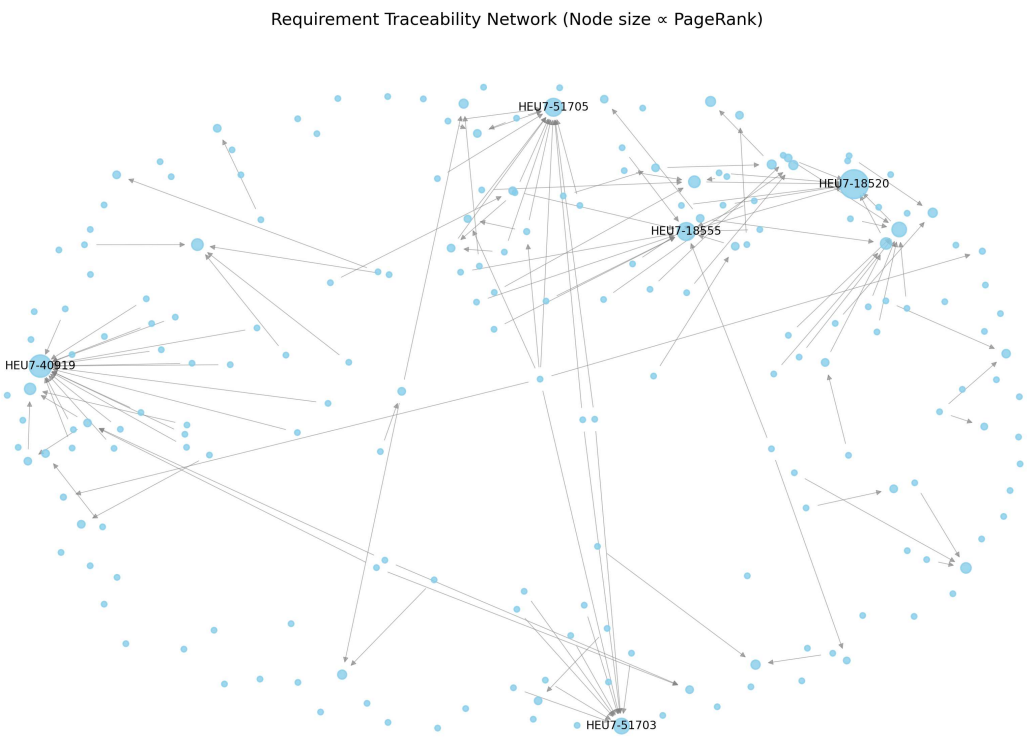

Figure 5.4: Graphical representation of the requirement dependency network.

Interpretation. This computational modeling stage provided a quantitative foundation for the theoretical analysis developed in the subsequent sections. By formally representing the requirement set as a directed network and computing its systemic indicators, it became possible to empirically characterize the structure of traceability in terms of depth, connectivity, and coupling. The results confirm that the observed system behaves as a *sparse and shallow dependency network*, consistent with theoretical expectations of modular architectures. The following section builds on these findings to interpret the functional meaning of different requirement roles and their implications for systems engineering theory.

5.2.4 Traceability as a Network

While a comprehensive Social Network Analysis (SNA) was not the primary focus, the matrix-based study and Python-derived metrics embody its fundamental principles, revealing the structural topology of traceability. In this formulation, in-degree and out-degree correspond to fundamental SNA metrics describing *centrality* and *influence*. Downstream-critical requirements represent nodes of high out-degree centrality, analogous to influencers in a communication

network: they broadcast design decisions. Upstream-sensitive nodes, conversely, have high in-degree centrality, behaving like information sinks that aggregate constraints. The coexistence of these patterns illustrates the system's hierarchical complexity, where requirements are not uniformly distributed but exhibit asymmetry typical of scale-free networks. From a theoretical standpoint, this resonates with **systems thinking** and **complex adaptive system theory**, where system behavior emerges from interactions among interconnected elements rather than from isolated components. The traceability network can therefore be seen as a *structural skeleton* of system knowledge: it reveals how design intent, constraints, and risks flow through the engineering hierarchy. Moreover, each link in the matrix embodies a potential *risk propagation path*. When a father requirement changes, all its descendants are potentially affected, introducing cascading effects similar to failure propagation in reliability networks. This perspective links directly to risk network theory in project management, as discussed by scholars applying SNA to model how project risks interrelate. In this context, traceability matrices provide empirical support for mapping the "network of influence" across design requirements, thus bridging the gap between engineering systems theory and project risk management.

5.2.5 Implications for Theoretical Modelling of Requirements

The classification of requirements into the four categories introduces a conceptual extension to traditional requirement traceability models. Conventional approaches often treat traceability links as binary relations — a link either exists or not. However, by analyzing the balance and directionality of those links, this work shows that requirements can assume functional roles within the system's architecture. This reinterpretation aligns with theoretical developments in *requirements meta-modeling* (Ramesh and Jarke, 2001) and traceability information networks (Gotel and Finkelstein, 1994), extending them toward a quantitative typology. For instance:

- Downstream-critical requirements correspond to what Ramesh called "high-rationale nodes," as they encode significant design decisions influencing many others.
- Upstream-sensitive nodes mirror "conformance nodes," which integrate multiple drivers and therefore constrain implementation flexibility.
- Equal-impact nodes fill the intermediate strata, maintaining balance and ensuring coherence.

- Independent nodes, if unintentional, highlight *traceability breaks*, which represent theoretical and practical gaps in requirement reasoning.

By quantifying these roles, the matrix transforms qualitative system architecture into measurable structural patterns. This contributes to theory by framing traceability not just as documentation, but as an emergent property of complex design networks. It allows theoretical propositions such as: "Projects exhibiting a high proportion of downstream-critical requirements will display higher change propagation risk," or "A system with excessive independent nodes indicates low traceability completeness." Such hypotheses, derived deductively from this case, can later be empirically tested across other projects or domains, extending the generalizability of the approach.

5.2.6 Methodological and Project Management Implications

From a methodological viewpoint, this analysis demonstrates how an **NLP-assisted traceability tool** can serve as the foundation for higher-order system analytics. By automating the identification of links, the tool enables the construction of large-scale adjacency matrices that would otherwise be infeasible manually — especially in industrial repositories exceeding 10,000 requirements. Once such a network is established, project managers can perform strategic evaluations:

1. Identify **hotspots** (requirements with high total degree) where verification and change control should be concentrated.

2. Detect **isolated clusters** that may represent disconnected subsystems or documentation omissions.

3. Monitor **network evolution** over time as a proxy for process maturity.

This represents a theoretical and practical contribution to project complexity management. By borrowing indicators from SNA (e.g., degree centrality, clustering, connectivity), requirement engineering gains the capacity to quantify complexity in a way that complements conventional metrics such as number of requirements or change rate. In essence, the matrix becomes a quantitative mirror of the system's organizational structure, reflecting how requirements interact, how teams communicate implicitly through dependencies, and how knowledge flows across the project. Building upon these considerations, the matrix also provides a valuable empirical

foundation for exploring change propagation and impact analysis, two central topics in modern project management. In complex engineering programs, change rarely remains confined to the component where it originates. As the *Cost of Change Curve* (Boehm, 1981) illustrates, the financial and organizational effort required to implement a change increases exponentially as the project moves from concept to validation and the presence of high out-degree nodes supports Boehm's intuition: design changes introduced late in the hierarchy entail exponential downstream rework. The traceability matrix serves as a predictive model for such behavior: requirements with multiple outgoing dependencies (>) effectively act as amplifiers of change cost. Each child requirement represents an additional verification path, an extra document to update, and possibly a new validation test to perform. By identifying these high-outdegree nodes early, the project manager can strategically allocate *contingency buffers*, prioritize documentation accuracy, and pre-approve secondary change pathways to reduce downstream disruption. Moreover, in a change-controlled environment such as automotive or aerospace development, where configuration management and requirement baselining are rigidly enforced, this network perspective aligns directly with the principles of **Integrated Change Control** (Project Management Institute, 2021). The downstream-critical requirements identified in the matrix can be treated as "change gateways": nodes that, if modified, should trigger broader impact analysis across the project. In this sense, the matrix operationalizes the theoretical model of systemic risk propagation by making interdependencies visible and measurable. The project manager can therefore move from *reactive impact analysis* (performed after a change request) to *proactive change anticipation*, predicting where potential disruptions are likely to occur and acting preventively. Equally important is the temporal and managerial dimension of these connections. Requirements traceability is not static; as projects evolve, new dependencies emerge, and old ones lose relevance. If the matrix is periodically updated and its evolution tracked, project managers can identify regions of the system where volatility—that is, the frequency of change—is consistently high. Such volatility hotspots are strongly correlated with managerial inefficiencies or ambiguities in the early definition of requirements. In project management terms, this aligns with scope creep analysis: understanding where and why requirements proliferate, merge, or diverge. Mapping these behaviors over time enables organizations to identify where additional managerial intervention, stakeholder alignment, or clarification of design intent is necessary.

From the perspective of risk and uncertainty management, the matrix can also be interpreted

as a risk propagation map. Each link represents a potential vector through which a design issue, requirement misinterpretation, or integration failure could spread across the system. This interpretation connects directly to the literature on Social Network Analysis (SNA) applied to project risk (e.g., (Delgado & Romero, 2016)), where nodes and edges correspond to risk carriers and their propagation channels. In this analogy, requirements are not just technical objects but risk transmission nodes. A highly connected requirement thus represents a systemic vulnerability—if misunderstood or delayed, it can cause cascading schedule or cost overruns across multiple teams. Conversely, a more modular structure, with limited cross-links between major subsystems, represents a resilient configuration, capable of absorbing localized disturbances without endangering overall project integrity. This theoretical lens further integrates with systems thinking and organizational resilience theory. By visualizing and quantifying the density of requirement relationships, managers can empirically measure the coupling between different subsystems. In project management, this coupling is often discussed in qualitative terms (e.g., "this subsystem depends heavily on that one"), but the traceability matrix translates it into quantitative evidence. This provides a concrete basis for deciding when to decouple development streams, redefine interface requirements, or modularize complex deliverables—all actions that contribute to reducing managerial complexity and improving schedule reliability.

Another critical dimension where the traceability matrix contributes to project management theory is knowledge coordination and resource allocation. In many product development organizations, responsibility for requirements is distributed among teams specializing in different domains (mechanical, electrical, software, testing, etc.). The matrix highlights the intersections of responsibility—requirements that depend on or refine those owned by other teams. These intersections often correspond to coordination bottlenecks, where communication lapses or ambiguity in ownership can lead to rework or conflicting decisions. From a resource management standpoint, understanding where these intersections occur allows project leaders to assign integration champions or cross-functional coordinators, ensuring that knowledge flow mirrors the structure of interdependencies revealed by the traceability network. This approach resonates strongly with concurrent engineering and integrated product development frameworks, where alignment of communication and technical dependency is a recognized success factor.

The cost implications of these insights are substantial. In practical terms, the cost of rework due to unanticipated requirement interactions can account for a significant fraction of total project effort—sometimes as high as 30–40% in large-scale systems projects (NASA, 2004).

The ability to identify high-impact nodes and dependency clusters enables targeted mitigation: investing verification resources where they yield the greatest reduction in rework risk. Moreover, this analytical approach supports progressive budget refinement: instead of assigning uniform contingency across all work packages, project managers can weight budgets according to dependency density, achieving a more efficient allocation of risk reserves.

From a strategic management standpoint, this use of requirement-level network analytics also contributes to the emerging discipline of Digital Thread Management. By linking system-level requirements, subsystem designs, and test artifacts, traceability becomes a continuous informational backbone that connects conceptual design to validation and operations. The insights drawn from the matrix—particularly those identifying highly connected or volatile requirements—can feed into predictive project dashboards, integrating technical and managerial metrics into a unified decision-support environment. This transition embodies the broader shift toward data-driven project management, where managerial decisions are informed not only by qualitative experience but by quantitative structural insights derived from traceability data.

Finally, by situating these findings within change management theory, the matrix helps refine our understanding of organizational adaptation. Change models such as Kotter's eight-step framework or PMI's change management guidelines emphasize the importance of communication and stakeholder alignment. Yet, they often lack a concrete mechanism to identify where communication should be concentrated. The traceability matrix fills this gap by revealing which requirements—and by extension, which teams—should be the focus of change communication. For example, a requirement with numerous downstream dependencies is not only technically critical but also communicatively central; any modification to it necessitates a structured communication cascade to ensure all affected stakeholders are informed. In this way, traceability becomes not just a technical asset but a managerial diagnostic tool, transforming abstract principles of change communication into actionable, data-driven strategies.

5.3 Contributions to the Practice

The findings of this research carry several direct implications for the practice of requirements engineering and project management within industrial environments, particularly those operating in large-scale, safety-critical, or heavily regulated sectors such as automotive and transportation. The developed semantic-based traceability tool demonstrates that it is possible to

substantially reduce the manual workload associated with the identification and maintenance of requirement hierarchies, while maintaining or even improving the accuracy of the resulting trace links. In practice, this translates into measurable time savings during the early design phases and into improved consistency in the propagation of requirement changes across development teams. Within organizations such as Iveco or other OEMs adopting tools like Polarion ALM, these benefits can directly impact project lead time, cost of quality, and traceability compliance, which are increasingly critical in the context of ISO 26262 and ASPICE standards.

A key contribution of this work is the demonstration that semantic similarity models — even those implemented through lightweight, pre-trained NLP architectures — can effectively support pre-requirement and inter-requirement traceability, two areas that have historically been underserved by commercial tools. The study shows that, by introducing a model capable of interpreting the meaning rather than the syntax of textual requirements, engineers can uncover latent links between functionally related but lexically dissimilar statements. This represents a fundamental evolution from keyword-based searches, which have traditionally dominated industrial practice. For practitioners, this means that the tool can serve as a "semantic assistant," automatically surfacing potential parent–child relationships and allowing engineers to focus on validation rather than discovery. The resulting process is not only faster, as quantified by the 25% productivity gain observed in testing, but also more robust to linguistic variability — an important factor in organizations where multiple teams, languages, or suppliers contribute to the requirement base.

From a procedural standpoint, the integration of such a tool within a requirements management workflow has several implications. First, it promotes a shift in the cognitive distribution of effort: engineers no longer need to rely solely on their memory or on navigational searches across hundreds of documents. Instead, the tool provides a ranked list of candidates whose semantic similarity scores guide their attention toward the most relevant portions of the repository. This new mode of operation enhances the traceability readiness of the organization by embedding a discovery mechanism directly into the authoring phase of requirements. Furthermore, by reducing the dependency on individual experience and tacit knowledge, the system contributes to standardizing the process of trace creation and reduces the risk of human oversight. In large-scale projects, where personnel rotation and document turnover are common, this can lead to significant gains in long-term maintainability and knowledge retention.

The industrial validation also highlights that the tool's value is not limited to productiv-

ity enhancement; it also supports process compliance and auditability. In regulated domains, demonstrating complete traceability between system, functional, and component-level requirements is a mandatory condition for certification. However, the manual construction of these traces is often fragmented and inconsistent. By automating the first layer of father–child discovery, the proposed system enables a more systematic approach to trace generation, creating a foundation of candidate links that can later be validated and formalized within the ALM platform. This ensures that even when engineers modify or extend requirements, potential links are suggested in real time, reducing the likelihood of non-compliance during audits. The ability to fine-tune the similarity threshold also gives practitioners control over the balance between coverage and precision, adapting the system to the rigor required by each project phase.

Another important contribution lies in the democratization of AI in requirements engineering. The prototype demonstrates that effective NLP-based retrieval can be achieved using openly available models, such as those provided through the Hugging Face ecosystem, without the need for proprietary or computationally intensive AI infrastructures. This lowers the entry barrier for organizations that may lack advanced data science teams but still wish to benefit from AI-driven traceability. It also paves the way for scalable deployment across departments: since the model operates on textual inputs and outputs simple ranked lists, it can be easily integrated with existing tools such as Polarion, Jama, or DOORS through API-based connectors. This architectural simplicity ensures that the system can be maintained and evolved without disrupting existing workflows — a crucial condition for adoption in industrial contexts where process stability and tool qualification are essential.

In practical terms, the insights gained from this research suggest several actionable recommendations for practitioners. First, requirement management teams should consider incorporating semantic retrieval as a complement to traditional link management modules in their ALM environments. Even partial adoption — for instance, using the tool to periodically verify and enrich traceability matrices — can yield measurable benefits in consistency and coverage. Second, organizations should establish domain-specific corpora to fine-tune the language model, as this would enhance its ability to capture specialized terminology and improve recall without sacrificing precision. Third, process engineers should view AI-based retrieval not merely as a tool but as a methodological enabler: one that allows teams to transition from reactive trace maintenance to proactive trace discovery. By embedding this approach into standard operating procedures, the creation of traceability links can evolve from a compliance task into a

continuous, data-supported process of knowledge integration.

Finally, the results hold strategic implications for project management and systems engineering governance. The reduction in manual effort implies that traceability maintenance can be performed more frequently and at finer granularity, allowing requirement dependencies to be updated dynamically as the project evolves. This can improve the synchronization between design, verification, and validation teams, leading to fewer inconsistencies in downstream activities such as testing or certification. Moreover, the feedback mechanisms observed during the experiment — where engineers interacted with the similarity scores to make informed judgments — suggest that such tools can also serve as decision-support systems, enhancing transparency and traceability reasoning across multidisciplinary teams.

In sum, this research provides both a technological and methodological contribution to the practice of requirements engineering. It demonstrates that even modest applications of NLP can deliver tangible benefits in speed, reliability, and standardization, while simultaneously reshaping how engineers engage with the traceability problem. The proposed approach is therefore not only a technical innovation but also a blueprint for integrating human expertise with AI-driven reasoning in the broader context of systems development.

Chapter 6

Conclusions

The research presented in this thesis demonstrates that the automation of requirement traceability through Natural Language Processing is not merely a technical convenience but a transformative enabler for systemic understanding in complex engineering projects. Starting from the development and validation of an NLP-based tool capable of suggesting hierarchical "father" requirements, the study has shown that artificial intelligence can significantly accelerate and enhance the analytical depth of requirement management processes. The quantitative validation confirmed tangible benefits in efficiency, with tool-assisted engineers outperforming manual traceability searches both in speed and coverage, thereby supporting the practical feasibility of semantic-based automation in industrial environments.

Beyond its empirical validation, the study extends its contribution to the theoretical modeling of requirements systems. By reconstructing the relational structure of an industrial repository into a traceability matrix and subsequently into a directed network, the research reframes traceability as an emergent property of a socio-technical system rather than as a static documentation activity. The network metrics and classifications derived from this analysis—such as in-degree, out-degree, and centrality—enable a structural interpretation of how information, constraints, and design intent propagate across the system. Requirements emerge as nodes within a complex web of interdependencies, where some act as critical broadcasters of change and others as absorbers of upstream constraints. This analytical perspective validates and extends the foundational insights of Gotel, Ramesh, and later scholars, transforming qualitative notions of traceability into quantifiable network phenomena.

From a project management perspective, the findings provide a bridge between systems engineering and organizational theory. The traceability matrix and derived metrics constitute

a quantitative language for describing project complexity, risk propagation, and communication pathways. High out-degree requirements were shown to correspond to potential cost-amplifying elements in the change process, while low-degree or isolated nodes highlighted areas of weak documentation or incomplete integration. Such insights align directly with established theories in change management and risk control, providing project leaders with actionable information to anticipate rather than react to disruptions. Moreover, by making interdependencies explicit, the network representation of traceability supports data-driven decision-making in configuration management, resource allocation, and cross-functional coordination, thereby operationalizing the principles of Integrated Change Control and digital-thread governance in practice.

On a broader theoretical level, this research contributes to the evolution of systems engineering toward a data-informed discipline. It demonstrates that the same dataset produced by an AI-assisted tool for operational purposes can serve as a foundation for higher-order reasoning about system behavior, organizational resilience, and knowledge flow. By visualizing and quantifying interconnections, managers and engineers can move from intuition-driven reasoning to evidence-based complexity management. The thesis thus positions requirement traceability at the intersection of engineering systems theory, network science, and project management, offering both a methodological contribution—through the implementation of NLP and network analytics—and a conceptual one—through the reframing of traceability as a living, dynamic representation of system coherence.

Ultimately, the work establishes a foundation for future research that could explore the temporal evolution of traceability networks, the correlation between network properties and project performance, and the influence of AI-driven automation on organizational structures. In doing so, it invites a rethinking of how requirement data is perceived: not as static records of compliance, but as the connective tissue through which complexity, risk, and innovation propagate in modern engineering enterprises.

6.1 Delimitations

This research was intentionally delimited in scope to ensure focus and feasibility within the timeframe and objectives of a master's thesis. The study concentrated specifically on the domain of inter-requirement traceability, excluding pre and post-requirement traceability ac-

tivities such as test linkage, verification, and validation processes. While these downstream phases are crucial in the broader lifecycle of systems engineering, they fall outside the core objective of this research, which was to explore the semantic relationships within the requirement corpus itself. This decision was motivated by both the identified gap in the literature—particularly the lack of mature methods for pre-requirement traceability, as emphasized by Gotel and Ramesh—and the desire to evaluate the feasibility of lightweight NLP-based approaches in addressing that gap.

Another deliberate delimitation concerned the operational environment chosen for experimentation. All data and validation activities were conducted using requirements extracted from the Polarion ALM environment, which is the tool currently adopted within Iveco's engineering ecosystem. This choice ensured contextual realism and relevance to industry practice but limited the study's generalizability to organizations using other requirement management systems such as IBM DOORS or Jama Connect. Although the underlying methodology is theoretically transferable, differences in data structure, metadata handling, and document granularity across platforms may influence the performance of the model and the ease of integration.

The research also restricted itself to a single natural language processing model, implemented through the Hugging Face environment. The model was used in its pre-trained configuration without domain-specific fine-tuning. This approach was chosen to demonstrate the baseline potential of semantic similarity algorithms even when employed "off-the-shelf." However, this delimitation implies that the model was not optimized for technical or domain-specific terminology, which could influence precision and recall in contexts where requirements contain highly specialized jargon. Future research could extend this work by experimenting with fine-tuned or custom-trained models, potentially improving the discrimination of nuanced linguistic relationships between requirements.

A further delimitation lies in the sample size and scope of validation. The controlled experiment involved a total of six engineers, divided evenly between a control group and an assisted group, each tasked with identifying hierarchical requirement links within a limited dataset. This setup provided a manageable yet representative basis for comparison but does not allow for large-scale statistical generalization. The study was not intended to produce population-wide behavioral insights but rather to provide an initial, evidence-based assessment of the tool's potential to enhance efficiency in real-world traceability tasks. Similarly, the dataset of requirements used for both the manual and automated analyses was confined to a single system

project. While this ensured consistency in content and terminology, it inherently narrows the contextual diversity of the tested material.

In addition, the research focused on specific performance indicators—namely, productivity gain, precision, recall, and success rate—while deliberately excluding other possible evaluation metrics such as usability, cognitive workload, or long-term learning effects. These aspects, although relevant to a holistic understanding of tool adoption, were beyond the temporal and methodological scope of the current study. Likewise, the user satisfaction assessment was exploratory rather than statistically validated, intended primarily to gather qualitative insights about usability rather than to establish definitive behavioral trends.

Finally, the tool architecture itself constitutes a delimitation. The system was developed as a standalone semantic analysis prototype operating on exported requirement data, rather than being fully integrated into the Polarion platform or tested in a live production environment. This design decision enabled flexible experimentation and rapid iteration but limits conclusions about scalability, data synchronization, and change management within an active ALM ecosystem. The emphasis was on demonstrating proof of concept rather than achieving enterprise-grade deployment readiness. Future extensions could focus on integrating the prototype directly into Polarion's API, expanding its scope to multi-document analysis, and testing its performance on large-scale repositories.

In sum, these delimitations were not weaknesses but necessary boundaries that ensured methodological clarity and focus. They allowed the research to target a well-defined and underexplored problem—semantic pre-requirement traceability—while maintaining control over experimental variables. The choices made reflect a conscious trade-off between depth and breadth, privileging a detailed, controlled evaluation of a novel approach over exhaustive coverage of the entire traceability landscape.

6.2 Limitations

While the study was carefully designed to maintain internal validity and methodological rigor, several limitations must be acknowledged. These limitations do not undermine the contribution of the research but rather highlight the contextual and technical constraints that inevitably accompany exploratory work on AI-assisted traceability in industrial environments.

A primary limitation concerns the **semantic model** employed in the tool. The Hugging

Face transformer was used in its general-purpose pre-trained configuration, without domain-specific adaptation or fine-tuning on engineering requirements. Although such large language models have demonstrated impressive generalization capabilities, their performance can degrade when exposed to highly technical vocabularies, acronyms, or implicit references typical of automotive or systems engineering documentation. This introduces an inevitable level of noise in similarity computations, particularly when two requirements share conceptual meaning but differ in syntactic expression. Consequently, the measured precision and recall may underestimate the true potential of a fine-tuned or custom-trained model specifically tailored for industrial traceability tasks.

A second limitation relates to the **ground truth data** used for evaluating the tool. The validation relied on existing links between requirements in Polarion that had been previously verified and approved by system engineers. While this ensured a reliable reference baseline, such human-established traceability is itself subject to error, inconsistency, and interpretive subjectivity. Engineers may have overlooked implicit relationships, or conversely, created links that reflect contextual understanding rather than strict textual evidence. As the tool evaluates similarity primarily on a linguistic level, discrepancies between semantic interpretation and engineering intent could affect measured accuracy. This limitation is particularly relevant when assessing "false positives" — links that may be linguistically plausible but absent from the manually validated dataset.

The **limited size and composition of the participant sample** in the productivity test also constrain the generalizability of the findings. The study involved only six engineers, divided equally between a control group and a tool-assisted group, all of whom belonged to the same organizational environment and shared similar backgrounds in requirements management. This homogeneity, while useful for reducing variability in experimental conditions, also limits the ability to generalize the observed improvement in performance to broader or more diverse engineering teams. Different levels of domain knowledge, familiarity with AI tools, or organizational practices could yield different outcomes in other settings.

Another limitation stems from the **restricted dataset size** used for evaluation. The test corpus consisted of a finite number of validated requirements extracted from a single Polarion project. Although this ensured terminological consistency and alignment of contextual semantics, it limited the tool's exposure to linguistic and structural diversity. Industrial requirements often vary significantly in length, complexity, and style across projects and domains, meaning

that a larger and more heterogeneous dataset could potentially reveal additional insights about the robustness and scalability of the proposed approach.

The study also faced constraints in **quantifying qualitative factors**, such as user experience and perceived trust in the AI-generated suggestions. While a satisfaction questionnaire was administered to capture initial impressions from tool users, the results were not statistically analyzed and remain anecdotal. As a result, the study could not draw definitive conclusions about cognitive workload reduction, learning curves, or acceptance barriers — all of which play a crucial role in the adoption of AI tools in engineering environments. Future work should incorporate mixed-method approaches, combining quantitative performance metrics with qualitative usability assessments to capture a more holistic picture of the tool's impact.

Furthermore, the **evaluation timeframe** constituted a limitation. Each engineer was allocated a one-hour session to complete the assigned traceability task. Although this setup was adequate for comparing manual versus tool-assisted performance, it may not fully reflect real-world usage, where engineers operate under varying degrees of time pressure, iterative revisions, and collaborative review cycles. The short experimental duration thus captures only a snapshot of the potential productivity gain, not its long-term sustainability or scalability across the lifecycle of large engineering projects.

Finally, certain **technical and infrastructural constraints** limited the fidelity of the tool's integration into the industrial workflow. The prototype operated as an external application that processed exported requirement data, rather than interacting directly with Polarion's live database through APIs. This architectural choice simplified testing but introduced latency in data synchronization and restricted the tool's capacity for continuous traceability maintenance — a key feature in production-grade systems. Moreover, the evaluation did not consider computational performance indicators such as response time, scalability under large repositories, or compatibility with other requirement management platforms. These factors will need to be addressed in future developments before the tool can be deployed operationally at scale.

In summary, these limitations reflect the inherent trade-offs of early-stage research positioned at the intersection of artificial intelligence and requirements engineering. They do not detract from the study's contributions but instead frame its findings within realistic boundaries. By recognizing the constraints imposed by dataset scope, model generality, and evaluation conditions, this research establishes a transparent foundation upon which future, more comprehensive investigations can build — including fine-tuning the model, scaling to multi-project

repositories, and integrating user-centered design principles for broader adoption.

6.3 Future Research Streams

The results of this research open multiple directions for future inquiry, both in the refinement of the developed tool and in the broader theoretical understanding of requirements traceability within complex engineering systems. First and foremost, further research should explore the **temporal evolution of traceability networks**. The present study offered a static snapshot of dependencies within a single project phase; however, projects are dynamic organisms where requirements evolve, merge, and branch over time. Longitudinal monitoring of network metrics — such as density, centrality, and clustering — could reveal how traceability structures mature or degrade as development progresses. Such temporal analysis would help identify early indicators of project instability, communication breakdown, or design divergence, providing valuable guidance for proactive project management interventions.

A second promising research direction involves the **integration of dynamic and probabilistic models** to complement the structural analysis presented here. Future work could combine network-based metrics with simulation techniques such as system dynamics or agent-based modeling to predict how changes in high-impact requirements propagate through the system. This would allow for quantitative estimation of the "ripple effect" of requirement modifications, bridging the gap between traceability analytics and cost or risk prediction models. Such an approach could also support sensitivity analyses to evaluate how specific requirement types (e.g., interface, safety, or performance-related) influence system robustness differently.

Another key area for exploration lies in the **validation and generalization of the NLP-based tool across domains and datasets**. While this thesis demonstrated the feasibility of using semantic similarity to identify hierarchical relations within an industrial automotive repository, future research could assess the model's adaptability to other engineering contexts — such as aerospace, rail, or software-intensive systems — where requirement formality and linguistic style differ significantly. Comparative studies could investigate whether domain-specific fine-tuning of the language model improves precision and recall, or whether hybrid approaches combining NLP with ontology-based reasoning yield superior results. Moreover, larger-scale benchmarking against proprietary tools like The Reuse Company's Requirements Quality Suite or Visure's Requirements ALM could provide empirical evidence of the relative strengths and

weaknesses of semantic versus rule-based approaches.

From a theoretical perspective, further studies could advance the conceptualization of **traceability as a complex adaptive network**. The classification introduced in this work — distinguishing downstream-critical, upstream-sensitive, equal-impact, and independent requirements — could be expanded by incorporating additional dimensions such as requirement criticality, verification status, or safety relevance. Integrating these attributes into network models would enable a multi-layered analysis, where functional, organizational, and temporal dependencies coexist. This could foster the emergence of a unified framework connecting systems engineering, social network theory, and organizational complexity, ultimately contributing to a more holistic theory of system interdependencies.

Future research should also consider **organizational and behavioral implications**. The use of AI-assisted traceability tools raises important questions regarding human trust, oversight, and decision-making. Studies could examine how engineers interact with automatically generated links, how confidence in AI suggestions evolves with experience, and what degree of transparency is necessary for adoption in safety-critical industries. This sociotechnical perspective would extend current work beyond technical validation, contributing to theories of digital transformation and human-AI collaboration in engineering design.

At the managerial level, an additional research opportunity lies in **linking traceability metrics to project performance indicators**, such as lead time, change request frequency, or rework effort. By correlating network properties (e.g., connectivity, modularity, or betweenness) with measurable project outcomes, scholars could empirically validate whether structural characteristics of requirement systems influence efficiency, quality, and cost. Such analyses would transform traceability from a compliance-oriented activity into a predictive management instrument — an evidence-based dashboard for complexity and risk control.

Finally, future investigations could address the **scalability and automation of traceability network generation**. The dataset used in this study comprised around 200 requirements from a single subsystem, whereas industrial repositories may contain tens of thousands. Research into computational optimization, parallel processing, and efficient visualization techniques could make large-scale, real-time traceability analytics feasible within enterprise environments. Combining these advances with the principles of the Digital Thread could lead to continuously updated traceability networks that reflect not only requirement dependencies but also links to design models, simulations, and test results — effectively closing the loop between

system definition and system validation.

In summary, future research should strive to expand both the **technical scope and the conceptual depth** of this work: evolving from static, tool-specific validation toward dynamic, cross-domain, and organizationally integrated models of traceability. By bridging AI, network science, and project management theory, the next generation of studies can transform traceability from an administrative necessity into a strategic capability for designing, managing, and evolving complex engineered systems.

References

Alexander, I. (2001). “Writing Better Requirements”. *IEEE Software*, 18:4, 19–21.

Antoniol, G. et al. (2002). “Recovering traceability links between code and documentation”. *IEEE Transactions on Software Engineering*, 28:10, 970–983. 10.1109/TSE.2002.1041053.

Barabási, A.-L. (2002). *Linked: The New Science of Networks*. Perseus Publishing.

Boehm, B. W. (1981). *Software Engineering Economics*. Prentice-Hall.

Checkland, P. (1999). *Systems Thinking, Systems Practice*. John Wiley & Sons.

Cleland-Huang, J. et al. (2002). “A Machine Learning Approach for Tracing Requirements to Code”. *Proceedings of the IEEE International Conference on Software Maintenance (ICSM’02)*. IEEE, 157–166. 10.1109/ICSM.2002.1167762.

Delgado, J. & Romero, D. (2016). “Analyzing project risk interdependencies using Social Network Analysis”. *International Journal of Project Management*, 34:8, 1579–1595.

Gotel, O. & Finkelstein, C. (1994). “An analysis of the requirements traceability problem”. *Proceedings of IEEE International Conference on Requirements Engineering*, 94–101.

Guo, J. L. et al. (2025). “Natural Language Processing for Requirements Traceability”. *Handbook on Natural Language Processing for Requirements Engineering*, 89–116. 10.1007/978-3-031-73143-3_4.

Hagberg, A. A. et al. (2008). “Exploring Network Structure, Dynamics, and Function using NetworkX”. *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 11–15.

Hunter, J. D. (2007). “Matplotlib: A 2D Graphics Environment”. *Computing in Science & Engineering*, 9:3, 90–95. 10.1109/MCSE.2007.55.

NASA (2004). *NASA Systems Engineering Handbook (SP-2007-6105 Rev1)*. Washington, D.C.: NASA Headquarters.

Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press.

Pinheiro, F. A. & Goguen, J. A. (1996). “An Object-Oriented Tool for Tracing Requirements”. *IEEE Software*, 13:3, 52–64. 10.1109/52.491460.

Pohl, K. (1996). *Process-Centered Requirements Engineering*. John Wiley & Sons.

Project Management Institute (2021). *A Guide to the Project Management Body of Knowledge (PMBOK®Guide) – Seventh Edition*. Newtown Square, PA: Project Management Institute.

Ramesh, B. & Jarke, M. (2001). "Toward reference models for requirements traceability". *IEEE Transactions on Software Engineering*, 27:1, 58–93. 10.1109/32.895989.

Ramesh, B. (1998). "Factors Influencing Requirements Traceability Practice". *Communications of the ACM*, 41:12, 37–44. 10.1145/290133.290138.

Spanoudakis, G. et al. (2004). "Rule-based generation of requirements traceability relations". *Journal of Systems and Software*, 72:2, 105–127. ISSN: 0164-1212.