Enabling the Collaborative Collection of Uncertainty Sources Regarding Confidentiality

Bachelor's Thesis of

Gabriel Gehrig

At the KIT Department of Informatics
KASTEL – Institute of Information Security and Dependability

First examiner:	PD Dr. Robert Heinrich
Second examiner:	Prof. Dr. Ralf Reussner
First advisor:	M.Sc. Sebastian Hahner
Second advisor:	M.Sc. Nicolas Boltz

03. July 2023 – 03. November 2023

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself. I have not used any other than the aids that I have mentioned. I have marked all parts of the thesis that I have included from referenced literature, either in their original wording or paraphrasing their contents. I have followed the by-laws to implement scientific integrity at KIT.

Karlsruhe, 03.11.2023

. . . . . .

(Gabriel Gehrig)

Abstract

With digitalization in progress, the amount of sensitive data stored in software systems is increasing. However, the confidentiality of this data can often not be guaranteed, as uncertainties with an impact on confidentiality exist, especially in the early stages of software development. As the consideration of uncertainties regarding confidentiality is still novel, there is a lack of awareness of the topic among software architects. Additionally, the existing knowledge is scattered among researchers and institutions, making it challenging to comprehend and utilize for software architects. Current research on uncertainties regarding confidentiality has focused on analyzing software systems to assess the possibilities of confidentiality violations, as well as the development of methods to classify uncertainties. However, these approaches are limited to the researchers' observed uncertainties, limiting the generalizability of classification systems, the validity of analysis results, and the development of mitigation strategies. This thesis presents an approach to enable the collection and management of knowledge on uncertainties regarding confidentiality, enabling software architects to comprehend better and identify uncertainties regarding confidentiality. Furthermore, the proposed approach strives to enable collaboration between researchers and practitioners to manage the effort to collect the knowledge and maintain it. To validate this approach, a prototype was developed and evaluated with a user study of 17 participants from software engineering, including 7 students, 5 researchers, and 5 practitioners. Results show that the approach can support software architects in identifying and describing uncertainties regarding confidentiality, even with limited prior knowledge, as they could identify and describe uncertainties correctly in a close-to-real-world scenario in 94.4% of the cases.

Zusammenfassung

Mit der zunemenden Digitalisierung nimmt die Menge gespeicherter sensibler Daten in Softwaresystemen zu. Jedoch kann die Vertraulichkeit dieser Daten in vielen Fällen nicht garantiert werden, da Ungewissheiten mit Auswirkung auf die Vertraulichkeit der Daten bestehen, insbesondere in den frühen Phasen der Softwareentwicklung. Da solche Ungewissheiten noch nicht ausreichend berücksichtigt werden und erforscht sind, besteht bei Softwarearchitekten ein Mangel an Bewusstsein für das Thema. Darüber hinaus ist das vorhandene Wissen über verschiedene Forscher und Institutionen verstreut, was es für Softwarearchitekten schwierig macht, das Wissen gesammelt zu erfassen und zu nutzen. Die aktuelle Forschung zu Ungewissheiten in Bezug auf Vertraulichkeit konzentriert sich auf die Analyse von Softwaresystemen, um die Möglichkeiten von Vertraulichkeitsverletzungen zu bewerten, sowie auf die Entwicklung von Methoden zur Klassifizierung von Ungewissheiten. Diese Ansätze beschränken sich jedoch auf die beobachteten Ungewissheiten der Forscher, was die Verallgemeinerbarkeit von Klassifikationssystemen, die Gültigkeit von Analysemethoden und die Entwicklung von Minderungsstrategien einschränkt. Diese Arbeit zielt darauf ab, zur Sammlung und Verwaltung von Wissen über Ungewissheiten in Bezug auf Vertraulichkeit beizutragen, um es SoftwarearchitektInnen zu ermöglichen, Ungewissheiten in Bezug auf Vertraulichkeit besser zu verstehen und diese in Ihren Software Architekturen zu identifizieren. Darüber hinaus soll der vorgeschlagene Ansatz die Zusammenarbeit zwischen Forschern und Praktikern ermöglichen, um den Aufwand für die Sammlung des Wissens möglichst gering zu halten. Um diesen Ansatz und seine Fähigkeit, die Forschungsziele zu erfüllen zu validieren, wurde ein Prototyp entwickelt und mit einer Nutzerstudie an 17 Teilnehmern aus dem Bereich Softwaretechnik evaluiert, darunter 7 Studenten, 5 Forscher und 5 Praktiker. Die Ergebnisse zeigen, dass der Ansatz Softwarearchitekten dabei unterstützen kann, Ungewissheiten in Bezug auf Vertraulichkeit zu identifizieren und zu beschreiben, auch bei Personen mit begrenztem Vorwissen, da sie in einer nahezu realen Umgebung Ungewissheiten in 94,4% der Fällen korrekt identifizieren und beschreiben konnten.

Contents

List of Figures

List of Tables

1 Introduction

This chapter establishes the context in which the approach of this thesis arises. First, Section 1.1 introduces the motivation and fundamental problem to which this thesis contributes. Then, Section 1.2 presents the research objectives and provides an overview of the intended contributions. Finally, Section 1.3 outlines the content and structure of the subsequent chapters.

1.1 Motivation and Problem Statement

As digitalization progresses, more tasks are handled digitally and automated. As more software systems are developed and deployed, software engineering has become a team effort. Today's software often consists of multiple components, also known as Component-Based Software Engineering (CBSE), developed by different teams. These components are reused in multiple software systems [41]. This development has led to software systems becoming complex. Furthermore, organizations are using software systems for more critical tasks. For example, Cyber-Physical Systems (CPS) used in the medical field or autonomous driving. As a result, such software systems need to uphold quality standards, especially reliability and security.

To support people in decision-making, software systems access stored data and process it to provide the user with information. As such, data creation, storage, and processing have become a critical part of software systems. When dealing with data, upholding the confidentiality level is a crucial issue. Confidentiality demands that "information is not made available or disclosed to unauthorized individuals, entities, or processes" [27]. Sensitive data such as personal information, medical records, and credit card information must be protected from unauthorized access. Non-sensitive data containing public, product, and metadata do not need to be protected. Upholding the confidentiality of sensitive data is not only a moral obligation for satisfied customers but also a legal obligation covered by the GDPR in Europe [39]. Consequently, there is a need to design software systems upholding the confidentiality of sensitive data throughout the system's lifetime. This task is also part of the role of the software architect, who is responsible for the design of the software system and builds the software's architecture. Software architecture is the structure of a software system, "which comprise software elements, the externally visible properties of those elements, and the relationships among them" [5]. Software architects face many challenges as the software is designed before it is implemented and deployed. With the possibility to use the final software system in many different environments,

input variables, user behavior, and other variables can vary, resulting in uncertainty for software architects. Uncertainty is "any departure from the unachievable ideal of complete determinism" [50]. Adaptive Software Systems, for example, are designed to adapt to changes in their environment, making it a key challenge for them to deal with uncertainty. From a data security perspective, this consequently means that there is always uncertainty regarding the structure, behavior, and environment of a software system during design and runtime [35] that can impact a software system's quality attributes, such as availability, reliability, and confidentiality.

A software architect is expected to deal with uncertainty by making assumptions about the environment and structure of the software system. However, the actual environment and structure of the software system runtime can overrule these assumptions. Thus, this can lead to a software system behavior that does not uphold the quality attributes. Consequently, recent research dedicated to the identification of uncertainties throughout the different stages of software development and the impact of these uncertainties on the software system's quality attributes [38, 17, 1, 25, 23]. In the context of confidentiality, not identifying uncertainties in software systems can lead to data breaches and data leaks, which can have severe consequences for the software system's users and owners, who do not uphold the law. Recently, the number of cyber-attacks and data breaches has increased significantly [11]. Thus, the need for software systems that uphold the confidentiality of sensitive data is more critical than ever.

Recent research that focuses on uncertainties with an impact on confidentiality has introduced methods to classify uncertainties and analyze software systems to assess the possibilities of confidentiality violations [25, 23]. However, these approaches are limited to the researchers' observed uncertainties, limiting the generalizability of classification systems, the validity of analysis methods, and the development of mitigation strategies. Consequently, there is a need to raise awareness on the topic of uncertainties regarding confidentiality and to collect observed uncertainties from researchers and practitioners. The need for collecting uncertainties is supported in the literature, as it describes the existence of a gap regarding the distribution of acquired knowledge to use for fostering a shared understanding [26]. Additionally, the literature describes the need for developing a "comprehensive understanding of the precise nature of uncertainty" [52]. Without it, developing mitigation strategies fringes on an ad-hoc, case-by-case basis [52]. Therefore, the collection of uncertainties regarding confidentiality is a crucial step to developing a comprehensive understanding of uncertainties regarding confidentiality. Furthermore, it can help to overcome the challenge of understanding the relation between uncertainties and confidentiality [22].

As one can only analyze what one knows, the collection of uncertainties regarding confidentiality can help to optimize analysis methods and develop mitigation strategies since software architects can use not only their uncertainty knowledge but also the uncertainty knowledge others have collected. This bachelor's thesis aims to address these issues and contribute to improving data confidentiality in software systems.

1.2 Research Questions and Contributions

This thesis divides its goals into two parts. The first goal involves enabling the possibility to collect the current state of knowledge on uncertainties regarding confidentiality, allowing the incorporation of new knowledge, and making it comprehensive to software architects. Comprehensibility, in the scope of this thesis, refers to the presentation of knowledge on uncertainties with an impact on data confidentiality in a manner that encompasses various aspects of uncertainty. It does not only focus on the understanding of uncertainties but also the ease of retrieving the information on uncertainties. This aspect of comprehensibility emphasizes easy access and retrieval of information without the need to include support for people with disabilities. Consequently, to collect knowledge on uncertainties and make it comprehensive, an approach is needed that allows for the description and relation of and between uncertainties regarding confidentiality. Additionally, this approach should allow for the possibility of extending the uncertainty collection with new uncertainty types to incorporate future findings on uncertainties regarding confidentiality and keep the knowledge on uncertainties up to date.

The second goal involves managing the effort to collect the knowledge, raise awareness, and increase understanding of uncertainties regarding confidentiality. While aiming to prepare uncertainty data for collection with the first goal, the second goal involves the endeavor to develop a collaborative approach to manage the effort to collect the knowledge and maintain it. This approach aims to allow for collecting knowledge across institutions and companies and the incorporation of future findings into the uncertainty collection.

The following research questions handle these goals and will be addressed throughout this thesis.

RQ1: How can the current state of knowledge on uncertainties regarding confidentiality be collected, managed, and made comprehensible for software architects?
The goal is to develop an approach for collecting and managing the knowledge on uncertainties regarding confidentiality, that is understandable and allows for the description and relation of and between uncertainties. Additionally, this approach should allow for a comprehensive uncertainty collection that software architects can use to learn about uncertainties regarding confidentiality and apply the knowledge to develop software systems with higher confidentiality standards.

RQ2: How can uncertainty knowledge be extended and maintained with manageable effort?
The goal is to develop an approach that allows for the extension of the uncertainty collection with new uncertainty types from researchers and practitioners and invite them to contribute to knowledge building. Additionally, this approach should strive to increase the awareness and understanding of uncertainties regarding confidentiality among software architects.

1.3 Outline

The subsequent chapters of this thesis aim to deliver answers to the stated questions. First, Chapter 2 presents the foundations of uncertainty and confidentiality. Chapter 3 gives an overview of related work that addresses this thesis topic. Then, Chapter 4 presents the approach for answering **RQ1**. Afterward, Chapter 5 extends the approach to answer **RQ2**. Finally, Chapter 6 presents the evaluation of the approach before Chapter 7 concludes this thesis by giving a summary of the thesis and an outlook on future work.

2 Foundations

This chapter gives an overview of the foundations required to understand the topic of this thesis. First, Section 2.1 gives an overview of uncertainty in software development. Then, Section 2.2 presents more details on uncertainties with an impact on confidentiality. Finally, Section 2.3 presents the concept of explainability.

2.1 Uncertainties in Software Engineering

The presence of uncertainties limits the validity of software architecture, as well as software systems, and poses a threat to successful software analysis, necessitating the mitigation of their impact. It is crucial to consider the existence of uncertainties and focus on developing methods to handle them effectively [16], to ensure successful software analysis. Previous research in software engineering has adopted this approach to deal with uncertainties systematically during the design phase [17]. However, before effectively dealing with uncertainties, they must be identified. A common approach for identification involves the usage of classifications and taxonomies to describe uncertainties and differentiate them from others. In the domain of model-based decision support systems, there exists a classification that categorizes uncertainties based on their *location*, "where the uncertainty manifests itself within the model complex" [50], *nature*, "whether the uncertainty is due to the imperfection of our knowledge or is due to the inherent variability of the phenomena being described" [50] and *level*, "where the uncertainty manifests itself along the spectrum between deterministic knowledge and total ignorance" [50]. These three dimensions describe the concept of uncertainties, according to the authors. Each dimension can exhibit predefined characteristics in itself. For instance, the *location* of uncertainty may reside within the context of the modeled system, representing its environment. Additionally, it may lie in the model itself, such as when using predefined variables to describe the model. A different approach to defining different types of uncertainties involves considering their *sources* [17]. In this study, uncertainties differ in their origin, including humans in the loop, learning, mobility, cyber-physical systems, and rapid evolution. These approaches identify various uncertainties and develop methods for managing uncertainties in self-adaptive systems during runtime [38]. As these approaches are specific to their field of software architecture, a classification of uncertainties intending to apply a broader range of models and software architecture types incorporates them [1].

2.2 Uncertainties with Impact on Confidentiality

In the domain of uncertainties within software systems affecting confidentiality, prior research employed a classification incorporating the dimensions of *Location*, *Architectural Element Type*, *Type*, *Manageability*, *Resolution Time*, *Impact on Confidentiality*, *Reducible by ADD*, and *Severity of Impact* [25]. They have derived their classification from existing classifications, adapting them to provide a more precise description of uncertainties with an impact on confidentiality. The *Location* describes "where uncertainty manifests itself within the architecture" [25] and can be either within the system structure, its environment, the system's behavior, or the input data. With the *Architectural Element Type*, the authors describe the architectural element affected by the uncertainty, such as a component, connector, interface, and hardware resource. The *Type* describes the nature of the uncertainty based on the existing knowledge of the uncertainty. Based on the authors' research, the uncertainty type can be Statistical Uncertainty, Scenario Uncertainty, or Recognized Ignorance. With *Manageability*, the authors describe the possibility of managing the uncertainty by acquiring more information or using appropriate means, having the options of Fully Reducible, Partially Reducible, or Irreducible. The category of *Resolution Time* describes the time at which the uncertainty can be resolved, either at Design Time, Runtime, Requirements Time, or Realization Time. *Reducible by ADD* describes the possibility of reducing uncertainty by making an architectural design decision, which can either be Yes or No. The *Impact on Confidentiality* describes the "potential impact on confidentiality requirements" [25] and can be either Direct, Indirect or None. Finally, *Severity of Impact* describes the severity of the impact on confidentiality in case of unresolved uncertainties, which can be either High, Low or None. With their classification evaluation, the authors presented a dataset of 40 uncertainties with an impact on confidentiality, which they could identify [24].

Building upon their previous work, the same group of researchers has developed a method for analyzing software architecture models to uncover violations in confidentiality when confronted with specific instances of uncertainty [23]. With their approach, they empower software architects to identify potential threats to confidentiality within their software systems during design time.

2.3 Explainability

Explainability in the context of software system behavior, is "the ability to provide human-interpretable explanations" [7]. Bersani et al. [7] present a 4-level framework to classify explainability, with the goal of achieving level four in explaining a phenomenon within a software system. A phenomenon within a software system is called an explanandum [7]. The first level describes *no explainability*, which describes the ignorance of an explanandum that needs an explanation. The second level is the *recognition of explainability needs*, the awareness of the existence of explanandum. On this level, a system tries to collect knowledge to explore the explainability of the explanandum. The third level is *local explainability*, which provides a first explanation of an explanandum in a specific context of the system behavior. This knowledge exists either in one place or pieces of knowledge in different places. The fourth level is *global explainability*, which explains an explanandum in various operating contexts. As this describes a road map to achieve explainability on phenomena in software system behavior, it can be applied to the explainability of uncertainties regarding confidentiality.

3 Related Work

This chapter gives an overview of existing research related to the topic of this work. The division of the related research falls into five categories: *Architectural Design Decisions and Uncertainty* (Section 3.1), *Risk Management* (Section 3.2), *Knowledge Management* (Section 3.3), *Collaboration in Software Engineering* (Section 3.4), *Software Architecture Modeling and Analysis* (Section 3.5), and *Collecting Uncertainties with an Impact on Confidentiality* (Section 3.6).

3.1 Design Decisions and Uncertainty

Handling uncertainties during software architectures' design implies the necessity for software architects to make fundamental design decisions. Therefore, software architectures inhabit the knowledge of these design decisions. Based on that fact, previous research has presented the idea of viewing software architecture "as a composition of a set of explicit design decisions" [29]. Moreover, their approach aims to help software architects understand underlying design decisions and establish dependencies between them. While their model can advise software architects when making design decisions, their research needs to improve in dealing with uncertainty. Furthermore, their concept does not provide tool support, limiting its potential for widespread utilization.

To address uncertainty, Garlan et al. [18] reuse a classification of uncertainties in *first-order* and *second-order* unknowns. With their classification, they provide a basis to differentiate uncertainties based on the level of knowledge about the uncertainty. *First-order* unknowns are uncertainties that are known to exist, and thus, mitigation strategies can be developed. *Second-order* unknowns are uncertainties unknown due to being outside of the scope and not being mitigated. While their objective of acknowledging the existence of uncertainties and aiming to identify them and develop mitigation strategies is important, their research does not present a solution for that challenge.

A meta-model that describes uncertainties more precisely can be found in the contribution of the OMG Standards Development Organization [53]. They introduce semantics for describing and measuring uncertainties in software architecture. As they aim to provide a standardized way of describing uncertainties and further enable the modeling of uncertainties in software architecture, their contribution to research assumes prior identification of uncertainties. Thus not providing a solution for the challenge of identifying uncertainties.

Approaches that aim to identify uncertainties in software architecture can be found in the work of Ramirez et al. [40] and Benkler [6]. By offering templates on how to identify and describe specific uncertainties, they contribute to a better understanding of these aspects among software architects. However, their approaches fall short of supporting software architects in making design decisions regarding uncertainties due to the narrow capabilities of describing uncertainties and difficulties in providing access to knowledge. Consequently, the impact of their findings on daily software development is limited.

Research covering the aspect of classifying uncertainties and providing support for software architects can be found in the work of Lupafya [33]. The author extends existing classifications of uncertainties in software architecture, like Garlan [18]. He proposes a classification of uncertainties that categorizes attributes in the dimensions: *descriptive attributes*, *source attributes*, *system attributes*, *manifestation attributes*, *time attributes*, and *mapping attributes*. The author also proposes a tool to make uncertainties visible in architectural models via an extension of a web application, as well as the option to download the architectural model for analysis purposes. However, as their approach helps in bringing awareness to the subject of uncertainties, it does not provide support for making design decisions regarding uncertainties. Furthermore, their tool support does not provide a concept for verifying new uncertainties, nor does it provide a concept for knowledge exchange on mitigation strategies. Moreover, their approach emphasizes quality attributes that do not include confidentiality.

Other research that offers tool support for software architects in making design decisions can be found in Zimmermann et al. [54] and Gerdes [20]. Zimmermann et al. [54] present a semi-automatic approach that focuses on reusing architectural design decisions for similar design problems. As their approach is a valuable contribution to the field of architectural design decisions, it does not include the aspect of uncertainty. Gerdes [20] introduce DecisionBuddy, a tool for supporting software architects in redesigning software architectures based on changed requirements. Their approach is based on the assumption of the existence of a software system and thus differentiates from the thesis' approach.

Jasser and Riebisch [30] introduce a completely different approach to making design decisions. They provide a repository for software architecture design decisions in which software architects can search for design decisions. They structured the repository based on classification with the ability to extend the repository with new design decisions. While their approach is a valuable contribution to the field of architectural design decisions, it does not include the aspect of uncertainty.

3.2 Risk Management

Risk management is a well-known concept beyond software engineering. It aims to identify, address, and eliminate software risks [9]. Boehm describes the fundamental concept of Risk Exposure as the probability of an unsatisfactory outcome multiplied by the loss from the event [9]. In addition, the author presents the six steps in software

risk management: *risk identification*, *risk analysis*, *risk prioritization*, *risk management planning*, *risk resolution*, and *risk monitoring* [9]. The author also lists software risk items, including mitigation techniques for each item. Research that picks up on the process of risk management is in the work of Van Scoy [46]. The author proposes to see a risk management cycle in the phases of *Identify*, *Analyze*, *Plan*, *Track*, and *Control*. Additionally, the author emphasizes the importance of communication in risk management. While these approaches focus on the process of risk management in general, they do not provide a concept for identifying uncertainties in software architecture. Their focus is on the performance, availability, and reliability of software systems but neglects the aspect of confidentiality. Their approaches are a top-level approach to risk management, while the approach in this thesis focuses on a more specific level. While their ideas on building a list of risk items and the importance of communication can contribute to identifying uncertainties, they do not provide a concept to extend the list. Without categorization, the list misses structure. Additionally, their view on communication differs from the one in this thesis, as this thesis focuses on communication between software architects to exchange knowledge and experiences and to verify or detect new uncertainties, while they focus on communication for documentation purposes.

An approach that includes security risks is in the work of Verdon and McGraw [48]. They detected risks based on the tier of the component in the software architecture. Additionally, they include risks in the dataflows between components. While their approach includes security risks and categorizes risks based on their location, their approach does not provide a concept for collecting and exchanging knowledge about risks and mitigation strategies, as well as taking into account the aspect of uncertainty.

Khan [31] presents a risk management framework in the cloud computing domain that includes the aspect of confidentiality. They propose a framework that includes the steps of *Identify*, *Analyze*, *Assess*, and *Action*. They categorized the risks and associated threats in the divisions *Availability*, *Confidentiality*, and *Integrity*. In addition, they associated priorities and probability with each threat. While their approach contains a template for identifying risks and threats, it does not include the concept of uncertainty or provide a concept for extending the list of risks and threats.

3.3 Knowledge Management

A contribution to creating value from knowledge through digital technologies can be found in the work of North et al. [37]. The authors propose a framework called “Knowledge Ladder 4.0”, which contains the steps: *Symbols*, *Data*, *Information*, *Knowledge*, *Actions*, *Competence*, and *Competitiveness*, based on the Knowledge Ladder [36]. The authors also present technologies that can support the different steps of the ladder. As such, they propose the usage of collaboration software to support the step from *Knowledge* to *Actions*. Because their approach focuses on value creation for organizations, it combines far more aspects than just knowledge management. While their concept of utilizing collaboration as

a support tool aligns with this thesis, it is too general to be applied to the specific domain of uncertainties in software architecture.

A subdomain of knowledge management contributing to knowledge sharing and collaboration includes Communities of Practice (COP) in knowledge management [28, 47, 12]. Community of Practice are "groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis" [51]. As one of the first to introduce the concept of COP to knowledge management, Iversion and McPhee [28] recognized that COPs could offer practical benefits for knowledge management. Venkatraman et al. [47] transfer the concept of COP to the digital world in knowledge management systems. They propose the Wiig Framework, which consists of the following dimensions: *Build Knowledge, Hold Knowledge, Pool Knowledge,* and *Use Knowledge.* They state that personal experience, education, training, media, and books build knowledge in people. Additionally, they express that people, documents, databases, and systems store the knowledge. Knowledge Management Systems (KMS) can be used to pool knowledge and make it accessible to others with brainstorming. The pooled knowledge can be used in work contexts and embedded in processes.

Choi et al. [12] conducted an empirical analysis of the impact of COPs on knowledge management systems on knowledge management activities and knowledge management performance and found that COPs have a positive impact on knowledge management and knowledge management systems [12]. Their approaches show the positive impact of COPs and collaboration on knowledge management but are focused on knowledge management on an organizational scale, while the approach in this thesis focuses on the specific domain of uncertainties in software architecture.

3.4 Collaboration in Software Engineering

On the topic of collaboration, most research in the domain of software architecture focuses on collaboration in software teams [32, 45, 3]. Layzell et al. [32] studied communication mechanisms within software development practitioner teams. They found that communication in software development is vital.

Tien et al. [45] developed a model for collaboration in software teams to meet the challenges of time pressure, budget, scope, and quality. They propose a cycle framework that consists of the following steps: *Communication/Requirements, Planning, Modelling, Construction,* and *Deployment.* In the stages of Planning and Construction, they propose the usage of collaboration tools to support the process.

A newer study by Bang et al. [3] interviewed software architects to get insights into their collaboration practices. They found that senior architects, who have the most experience and knowledge, take only a limited participation in the detailed design. Thus, junior architects are left to make design decisions independently while having less experience and a more significant number of tasks to complete. While these approaches give more

insights into the collaboration practices of software architects. They focus on the practices inside organizational borders.

Other research that emphasizes the importance of collaboration can be found in the work of de Vreede et al. [49] and Sterz et al. [43, 44]. De Vreede et al. [49] ask for more research on collaboration in software teams and organizations. Sterz et al. [43, 44] investigate the allocation of responsibilities for upholding standards regarding confidentiality in the realm of today's software systems. Their approach emphasizes continuous collaboration and cooperation among companies developing a software system to clarify responsibilities and enhance security measures. Although not directly applicable, this approach arguably holds considerable advantages in enhancing the understanding of uncertainties and reducing their effects. Consequently, the aim is to modify this approach to align with the goals of this thesis.

3.5 Software Architecture Modeling and Analysis Support

Software Architecture Modeling and Analysis is a broad field in software engineering. As such, this section focuses on the aspects of modeling and analysis support on software architecture for software architects.

Support, which combines both aspects, can be found in the work of Reussner et al. [41]. Their software, Palladio, allows them to reveal performance and reliability issues in software architectures based on the Palladio Component Model (PCM). While their approach gives insights into the performance and reliability of software architectures, it does not include the aspect of confidentiality. In addition to that, their approach provides analysis capabilities but does not provide a concept for identifying uncertainties.

An approach that supports those capabilities can be found in the work of Babar et al. [2]. The authors introduce PAKME, a tool for capturing a broad range knowledge of software architectural knowledge with additional capabilities of searching for specific knowledge as well as evaluation capabilities. Quiver by Gopalakrishnan et al. [21] supports architectural decisions during the design process by providing architectural knowledge and recommending architectures based on the most critical quality attributes. As these approaches provide valuable contributions to the field of architectural analysis, they do not include the aspect of uncertainty.

Colesky et al. [13] introduced their approach to developing privacy patterns, which are reusable software designs for reoccurring software engineering problems on privacy. To use their models, they developed an approach that allows software architects to access, collaborate, and share their knowledge about privacy patterns [14]. While their approach includes collaboration and knowledge sharing, it is designed for privacy patterns and does not include the aspect of uncertainty.

Approaches that include the aspect of uncertainties can be found in the work of Lytra and Zdun [34] and Esfahani et al. [15]. Lytra and Zdun [34] use fuzzy logic to analyze

uncertainties and support software architects by providing reusable architecture design decisions. Their approach utilizes a semi-automatic approach with limitations on making architectural design decisions. GuideArch by Esfahani et al. [15] uses fuzzy logic as well. They map different architectural design decisions to quality attributes to support software architects in making the correct design decisions based on their priorities. While both approaches include uncertainty, they do not involve confidentiality as a quality attribute.

As seen, several approaches support software architects through analysis methods and modeling. As none of them contribute to considering uncertainties regarding confidentiality when analyzing and modeling software architectures, revealing a need for such an approach, which this thesis aims to address.

3.6 Collecting Uncertainties with Impact on Confidentiality

As far as current knowledge extends, an existing approach for collecting knowledge on confidentiality uncertainties has yet to be identified. However, it is described in the literature that there is a need for collecting knowledge on uncertainties with an impact on confidentiality, as there remains a gap regarding the distribution of acquired knowledge to use for fostering a shared understanding of that subject matter [26] and to identify additional uncertainties. Insufficient distribution and collaboration impede identifying uncertainties, yet unknown to the community. Furthermore, concrete uncertainties are essential requirements for the practical usage of analysis methods, as findings from analysis methods are limited to known uncertainties and their impact [23]. Research has demonstrated the existence of diverse knowledge and opinions regarding uncertainties, while all agree on the importance of addressing uncertainty [26]. Thus, to address the gap in the distribution of knowledge on uncertainties with an impact on confidentiality, foster a shared understanding of that subject matter, and making analysis methods more effective, collecting knowledge on uncertainties with an impact on confidentiality is needed.

4 Capturing Uncertainties Regarding Confidentiality

This chapter introduces the methodology for capturing knowledge of uncertainties affecting confidentiality and lays the groundwork for the collection of uncertainties. Section 4.1 presents the running example referenced in the subsequent chapters. Section 4.2 introduces the procedural structure designed to achieve the research objectives. Next, Section 4.3 presents the meta-model for mapping uncertainties impacting confidentiality, while Section 4.4 outlines the data preparation approach for collecting uncertainties. Lastly, Section 4.5 summarizes the key elements discussed in this chapter.

As discussed in Chapter 2 and Chapter 3, research has acknowledged the issues of uncertainties software architects are faced with and have developed methods to classify uncertainties and quantify and minimize their impact. However, Chapter 3 also showed that there needs to be more research regarding the capturing of knowledge on uncertainties in software architecture that can have an impact on data confidentiality. This knowledge is essential to increase the widespread understanding of such uncertainties and thus allow software architects to identify them in their architectures. As research in the domain of risk management has shown, the identification of risks is a crucial step before mitigating them [46]. Thus, capturing knowledge regarding uncertainties is an essential step before analyzing them through analysis methods like Hahner et al. [23] developed.

4.1 Running Example

To explain the following concept, Figure 4.1 illustrates the example of an *Online Shop*. It describes a scenario in which a software architect faces designing an *Online Shop*. During the design phase, the architect is aware of the utilization of an *On Premise Server* and a *Database Service* as fundamental components of the infrastructure. Within this simple structure, incoming customer requests for product information need to be handled by the *Database Service*. Furthermore, it manages personal information on checkout, including delivery address and payment details. While these tasks of the *Database Service* are already straightforward during the design phase, the architect has limited knowledge of the particular manner the *Database Service* provider uses to operate. As a result, the software architect faces different uncertainties. This can have an impact on multiple aspects of the software system. For example, regarding availability, the architect has to assess the probability of availability outages of the *Database Service*. As far as confidentiality is

concerned, the software architect cannot determine with certainty the secure processing and storage of information, especially sensitive data.

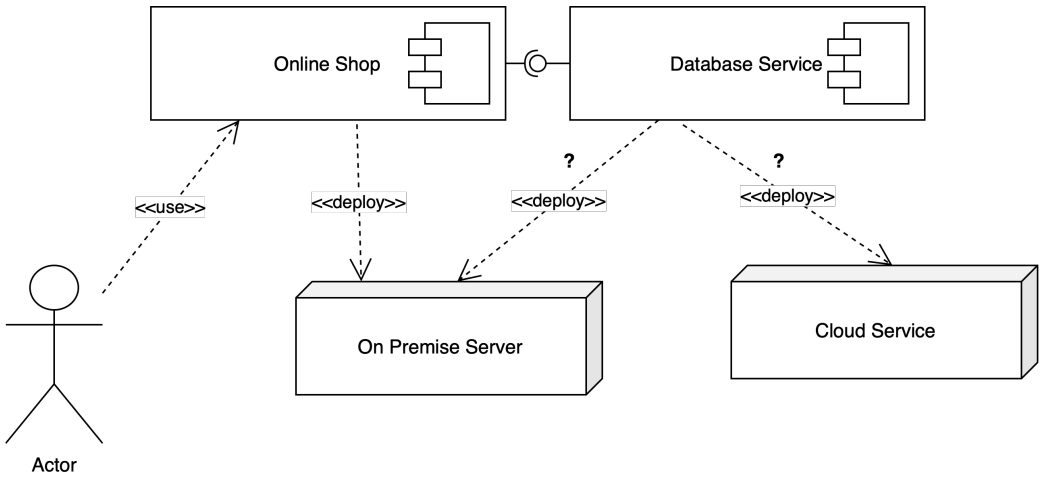


Figure 4.1: An architecture model of a simple online shop

Naturally, a software architect faced with uncertainties will be forced to rely on assumptions and make decisions to the best of his or her knowledge. As experience can help software architects in making such decisions and minimizing uncertainties, there always remains a risk of software failures. In this case, the unavailability of the service and unlawful processing and saving of data. This example shows, that uncertainties exist in simple software architectures with us stating only a few of them. Therefore, it is assumable, that the amount of uncertainties in complex software systems is significant.

4.2 Structure and Requirements

The approach addressing research questions **RQ1** and **RQ2** includes the following requirements for answering the research questions before it proposes visions to fulfill them.

The following list presents an overview of the requirements that were defined:

R1: Utilization of a clear and comprehensible classification system to describe and structure uncertainties

R2: Giving an overview of all collected uncertainties and their properties

R3: Stating information about the specific classification terminology, including the different uncertainty types and options

R4: Giving context to each uncertainty, including a visualization to show different levels of abstraction

R5: Enable hierarchical relationship structures between uncertainties

R6: Enable categorization by keywords

R7: Allowing filtering and sorting mechanisms for the uncertainties

R8: Allowing search mechanisms for the uncertainties

R9: Allow cross-referencing between uncertainties and classification information

These requirements are the result of the following observations. Firstly, using an existing classification system helps in structuring the data and makes it easier to understand (**R1**), especially for inexperienced software engineers and software engineers with little knowledge of uncertainties. Secondly, to motivate software engineers to mitigate uncertainties and make them aware of the importance of doing so, making clear the advantages of mitigating uncertainty in the early stages of the software development process is essential. It helps in the development of a deeper understanding of uncertainties. The advantages include that it gets more expensive to mitigate uncertainties the later they are discovered [8]. Thirdly, to increase the understanding of uncertainties, knowledge needs to be comprehensive. Thus, giving an overview of all uncertainties and their attributes (**R2**) is essential. Additionally, this can help prevent the creation of duplicate uncertainties. Fourthly, the current knowledge regarding uncertainties lacks a shared understanding and terminology. This missing shared understanding can be observed in using different terminology in classifications regarding uncertainty, such as the terminologies of Hahner et al. [25] and Ramirez et al. [40]. Consequently, the different types of uncertainties, their manifestations, and properties (**R3**) can significantly benefit the description of specific classification terms.

Another way of making clear the differences between uncertainties, in addition to category options, is to give context to each uncertainty (**R4**). By adding descriptions, examples, or even better visualizations, such as graphs or software architectural models to each uncertainty, the understanding of the different abstraction levels and manifestations of specific properties can be increased. This can be advantageous, especially for software engineers with no or little prior knowledge. Enabling hierarchical relationship structures between uncertainties (**R5**) can significantly benefit improving knowledge organization. Not only can it help in determining if an existing uncertainty already covers a new uncertainty and thus favors the incorporation of new uncertainties over the creation of duplicate uncertainties, but it can also help in increasing the understanding of uncertainties. For example, when taking the presented running example from Section 4.1, there persists an uncertainty regarding user behavior. This uncertainty contains multiple dimensions, e.g., regarding the type, correctness, or validation of data the user provides before the data transfers over to the online shop. While these dimensions pertain to user behavior, they focus on different aspects. Consequently, one should treat these uncertainties differently. Including hierarchical relationship structures in the knowledge capture of uncertainties can help in connecting these uncertainties and thus increase the understanding of uncertainties as well as the organization of knowledge. Additionally, categorizing uncertainties not only by classification properties but also by keywords (**R6**) can help in organizing knowledge. This benefit is observable through the running example from Section 4.1 when taking the uncertainty regarding the trustworthiness of the database provider for the online shop.

At design time, the software architect might need help to determine with certainty the secure processing and storage of data. Another uncertainty that might exist regarding trustworthiness is the trustworthiness of the user input. The software architect might need help to determine with certainty if the user provides validated data before being processed in the online shop. As those uncertainties differ, both aim to determine the level of trust. As classification properties might not be able to capture this similarity, using keywords such as "trust" might improve the organization of knowledge to determine where other uncertainties fit in. Furthermore, being able to filter through different uncertainties and be able to sort them by specific aspects (**R7**), such as manifestations and properties, can allow software engineers to get a better understanding of the different manifestations and properties of specific uncertainties.

Another way of making knowledge more comprehensive is to allow search mechanisms for the uncertainties (**R8**). Additionally, these requirements show a correlation between the goals of increasing the understanding of uncertainties (**RQ1**) and organizing knowledge that enables the incorporation of new uncertainties and specification of known uncertainties (**RQ2**). Allowing cross-referencing between uncertainties and classification information (**R9**) is another example of this correlation. Through cross-referencing, software engineers can get information about the underlying classification while reading about a specific uncertainty. Additionally, it can help organize knowledge by allowing software engineers to determine if an existing uncertainty already covers their uncertainty or if it is new and how it can be specified using the classification terminology.

4.3 Meta Model of Uncertainty Description

For describing uncertainties with an impact on confidentiality, this approach contains a meta-model. As shown in Figure 4.2, the meta-model consists of the uncertainty attributes: *ID*, *Name*, *Definition*, *Example*, *Keywords*, *ClassificationProperty*. Additionally, Figure 4.3 shows the meta-model as an *Entity-Relationship*-Model, which allows for a better understanding of how the data on uncertainties may be structured to store it in a database.

Each uncertainty is distinguished by a unique *ID*, enabling reference to that specific uncertainty. The *Name* offers a concise depiction, while the *Definition* provides a more elaborate explanation. An *Example* serves as an illustrative instance of the uncertainty in question. *Keywords* aid in categorizing the uncertainty, while the *ClassificationProperty* attribute allows for classification by a system that assesses uncertainties affecting confidentiality. The *related to* relationship establishes connections with other uncertainties that share associations with the specific uncertainty. Consequently, leaving the *ClassificationProperty* as general as possible allows for the incorporation of new classification systems at a later point, as well as adding new properties to the existing classification system and adding more flexibility and extensibility to the meta-model. For the scope of this thesis, the approach uses the classification system developed by Hahner et al. [25] to meet the requirement of utilizing a clear and comprehensive classification system (**R1**). Figure 4.4

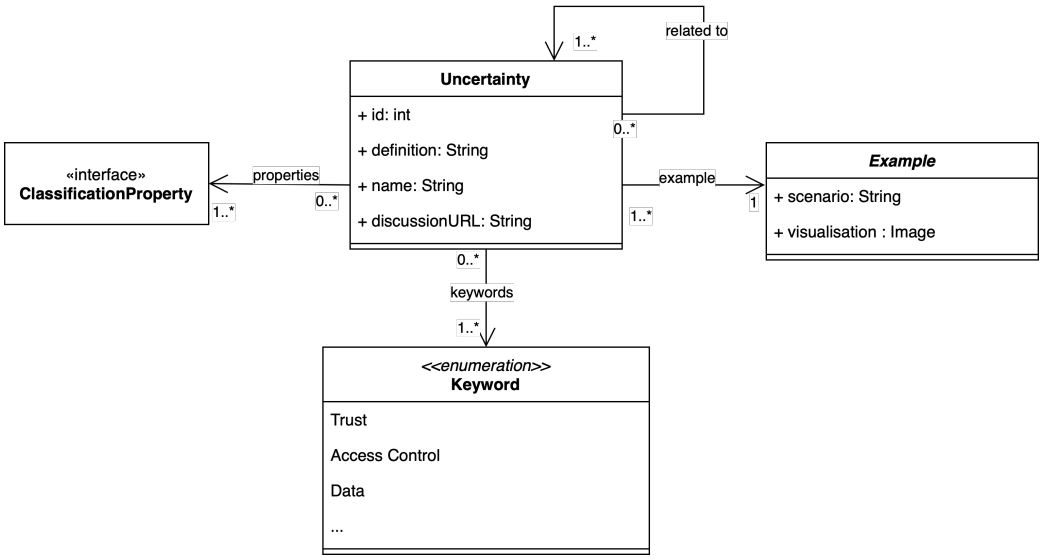


Figure 4.2: Meta model for uncertainty and its properties

illustrates an individual model of that, including the classification system by Hahner et al. [25].

4.4 Explainability of Uncertainties

Section 2.3 introduced the concept of explainability in the context of software systems. It described the four levels of explainability, intending to achieve level four in explaining a phenomenon within a software system. Based on the authors' framework [7], to support explainability for uncertainties regarding confidentiality, it is critical to raise awareness of uncertainties, collect knowledge, and centralize the management of uncertainties as various software architects have only a partial view of that subject matter. Thus, this approach emphasizes explaining uncertainties in a way that goes beyond a simple table of uncertainties, to allow for the exploration of explainability (Level 2). While it is essential to have a table of uncertainties, as it allows for a quick overview of all uncertainties **R2**, it needs more details for a software architect with little knowledge of uncertainties to understand the different aspects of uncertainties. Figure 4.5 shows the difference between the descriptive capabilities. It illustrates uncertainty in a table view and detail view from the running example in Section 4.1 that relates to the trustworthiness of the database provider for the online shop. While the simple listing only states the *ID*, *Name*, and the different *ClassificationProperties*, the detailed listing allows more in-depth information, like *Definition*, *Example*, *Related*, and *Keywords* attributes, as well as a visual representation of the uncertainty, which relates to **R4**, giving context, and **R6**, using keywords. Moreover, the detailed listing enables for associating uncertainties with each other. This association of uncertainties creates a hierarchical relationship structure between uncertainties **R5**,

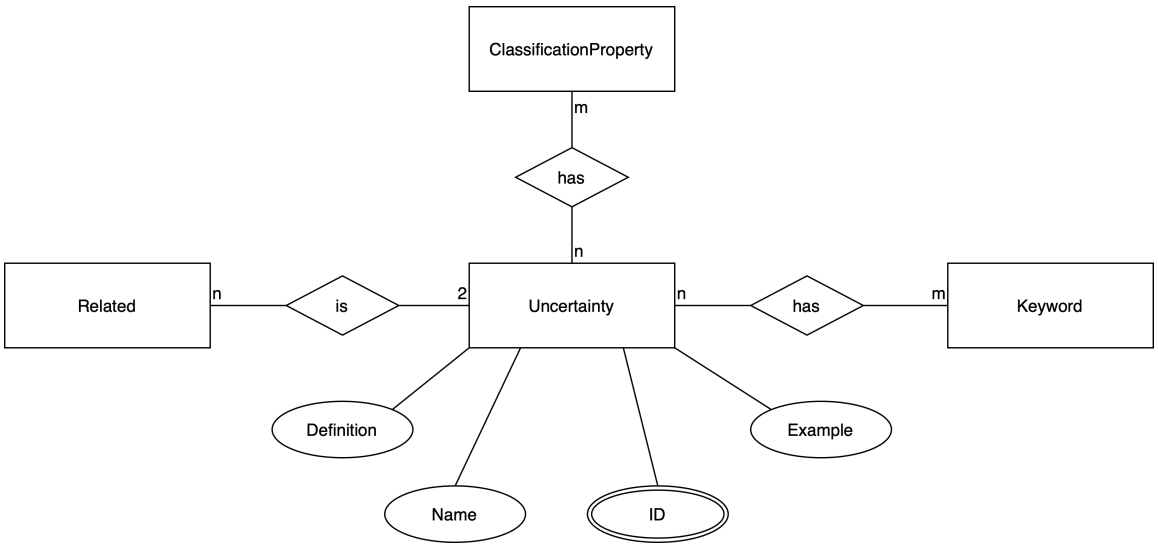


Figure 4.3: ER-Model of meta model for uncertainty and its properties

illustrated in Figure 4.6. It introduces another dimension of possibilities for a deeper understanding of uncertainties.

Similarly, information on the classification system can be displayed. Figure 4.7 illustrates this by using the classification category "Location" from the classification system developed by Hahner et al. [25]. While the simple listing only states the *Name*, *Description* and *Options*, the detailed listing can utilize a visual representation of the classification category to make the differences between the different options clear and meet **R3**, which requires to give information on the used terminology.

4.5 Summary

This chapter introduced requirements for addressing the research questions (**RQ1** and **RQ2**). It presented an approach to capture the knowledge of uncertainties with an impact on confidentiality (**RQ1**). This approach included the development of a meta-model in Section 4.3, which allows for a classification system to classify uncertainties (**R1**), including multiple attributes that allow for the description of uncertainties. Section 4.4 addressed the requirements **R2**, giving an overview on uncertainties, **R3**, stating information on the terminology, **R4**, give context, **R5**, represent relationships between uncertainties, and **R6**, use keywords. This chapter demonstrated that fulfilling these requirements individually is achievable through a more detailed description of uncertainties. To answer **RQ1** completely, the information must be provided for software architects to retrieve. This chapter did not address **RQ2** and the requirements **R7**, **R8** and **R9**.

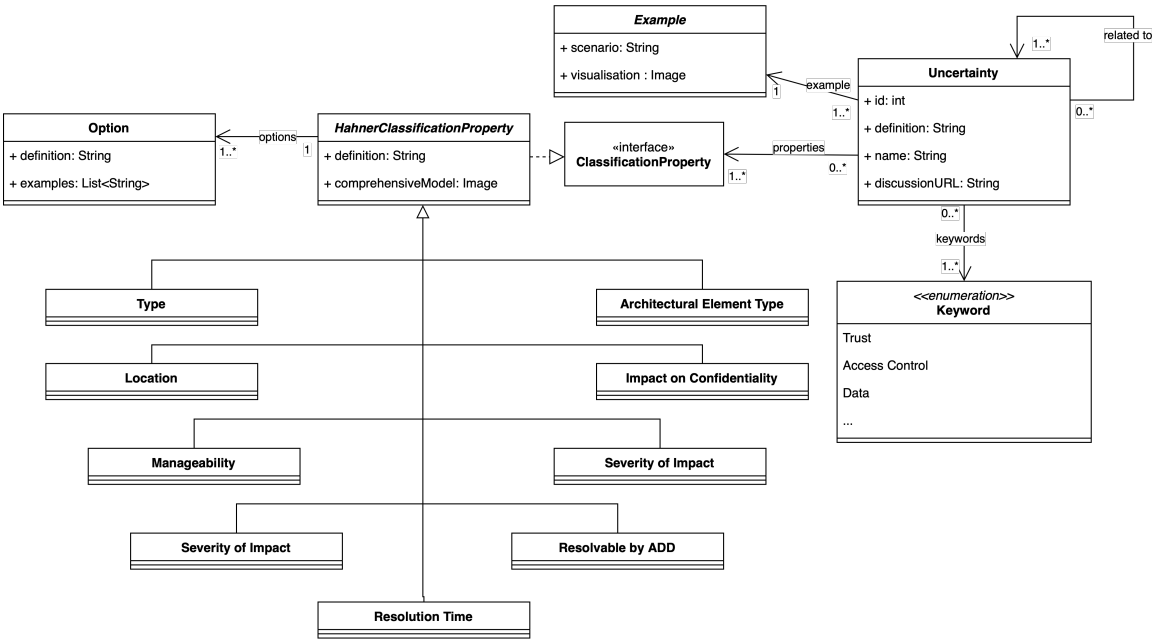


Figure 4.4: Class diagram modeling uncertainty when using the classification system of Hahner et al. [25]

ID	Name	Keywords	Location	Architectural Element Type	Type	Manageability	Reducible by ADD	Resolution Time	Impact on Confidentiality	Severity of Impact
15	Is the components provider trustworthy?	Trust	System Environment	Component	Recognized Ignorance	Irreducible	Not Reducible	Design Time	Direct	High

Simple table listing of uncertainty

15 - Is the components provider trustworthy	
Concerns the system's environment and signifies recognized ignorance about the trustworthiness of the provider of a component	
Keywords: Trust	
Location (Describes where an uncertainty manifests itself within the architecture): System Environment (System's context, including hardware resources and the external situation)	Type (How much is known about the uncertainty and how it can be described): Recognized Ignorance (Awareness of the uncertainty but no mitigation or description strategy is in place)
AET (Elements to which an uncertainty can be assigned): Component (Assignable to software components)	Manageability (Can more knowledge or appropriate means reduce the uncertainty): Irreducible (Uncertainty cannot be further reduced)
Reducible by ADD (Uncertainty resolvable by an architectural design decision): Not Reducible (Uncertainty is not resolvable or treatable by taking an ADD)	Impact on Confidentiality (Potential impact on confidentiality requirements): Direct (Direct impact on confidentiality)
Resolution Time (Time at which the uncertainty is expected to be fully resolved): Design Time (As soon as the system is designed, the uncertainty is resolved)	Severity of Impact (Describes the severity if uncertainty is not mitigated): High (Total loss of confidentiality, or sensitive data)
Example: Within a mobility system, a company might offer Vehicle rentals. A user can rent a vehicle via a mobile app. The company then collects all necessary information and saves it in a database by sending the data to the database service. The company decided that they don't want to host their own database, but rather use a database service provider. The company has to decide on a database service provider with keeping in mind that they have to follow EU law, thus their data has to be stored in Europe.	

Detailed Listing of Uncertainty

Figure 4.5: Simple listing vs detailed listing of uncertainty

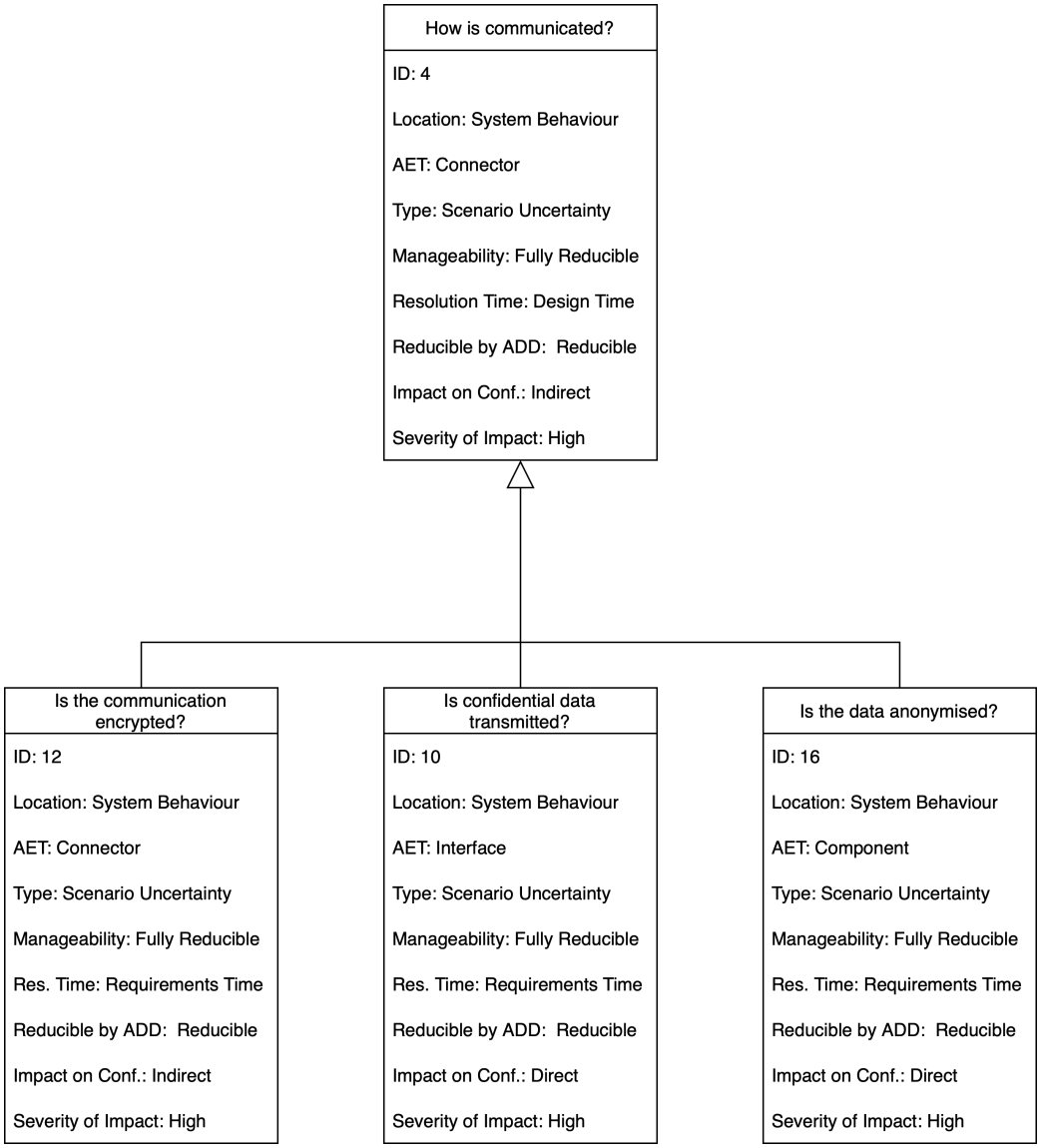
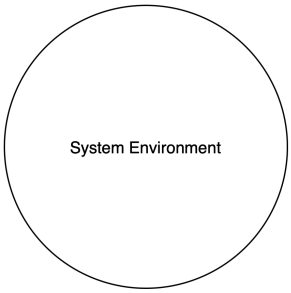
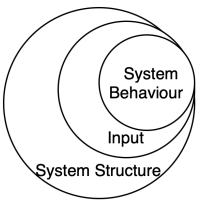


Figure 4.6: The hierarchical relationship of Uncertainty “How is Communicated”

Category	Description	Option 1	Option 2	Option 3	Option 4
Location	Describes where an uncertainty manifests itself within the architecture.	Input	System Behaviour	System Environment	System Structure

Simple table listing of category “Location”

Category Location	
Description: Describes where an uncertainty manifests itself within the architecture.	
Input: Input provided by external actors For example: People using the software	System Behaviour: Behaviour of the system and its components as well as their communication For example: The handling of sensitive data within a component.
System Environment: System's context, including hardware resources and the external situation For example: The physical location of a database.	Structure: The Structure of the system For example: Components and their static wiring, assembly, and allocation
	

Detailed listing of category “Location”

Figure 4.7: Simple listing vs detailed listing of classification category “Location”

5 Managing Uncertainties by Collaboration

This chapter, presents an approach to addressing **RQ2**, which asks for the extendability and manageability of the knowledge of uncertainties. Section 5.1 provides an overview of utilizing the knowledge structured in Chapter 4. Subsequently, Section 5.2 discusses the requirements for ensuring extensibility and maintainability of the knowledge. Section 5.3 introduces a collaborative approach to addressing these aspects. Furthermore, Section 5.4 outlines the necessities for realizing the approach, along with the presentation of a prototype. Finally, Section 5.5 summarizes the chapter's achievements.

5.1 Using the Knowledge on Uncertainties

The existence of knowledge on uncertainties related to confidentiality needs to be improved for software architects to identify uncertainties effectively. It requires clarity and comprehensibility, emphasizing the need for data preparation, including data structuring. Improving user-friendliness by providing an interface for retrieving information can enhance clarity and comprehensibility. Software architects should retrieve information on uncertainties via the interface, regardless of location. Additionally, the user interface should allow for the filtering of uncertainties by different properties (**R7**), as well as the possibility of searching for a specific uncertainty (**R8**). In doing so, the approach can support users in identifying uncertainties that are relevant to them and their software architecture. Moreover, considering the various methods through which individuals acquire and comprehend information, incorporating diverse approaches to retrieve data on uncertainties contributes to an overall comprehension of the collection. In combination with the explainability covered in Section 4.4, software architects can use the knowledge to identify possible confidentiality violations in their software architectures, analyze them, and develop mitigation strategies.

5.2 Extending the Uncertainty Collection

The collection needs to be extendable to be able to incorporate new knowledge and to keep the collection up to date. A meta-model was presented in Section 4.3 that allows for the collection extension. For example, the uncertainty *Is the data anonymized?* and *How is communicated?* from the dataset [24] can be added to the collection as new uncertainties (Figure 5.1). However, while the meta-model allows for the extension of the collection,

other aspects need to cover introducing a system to manage and store the actual data on the uncertainties.

16 - Is the data anonymised	
Relates to uncertainty regarding whether data is anonymized by a component within the architecture.	
Keywords: Data	
Location (Describes where an uncertainty manifests itself within the architecture): System Behaviour (Behaviour of the system and its components as well as their communication)	Type (How much is known about the uncertainty and how it can be described): Scenario Uncertainty (Distinct scenarios depending on the uncertain outcome, no statistical means.)
AET (Elements to which an uncertainty can be assigned): Component (Assignable to software components)	Manageability (Can more knowledge or appropriate means reduce the uncertainty): Fully Reducible (Reducible)
Reducible by ADD (Uncertainty resolvable by an architectural design decision): Reducible (Uncertainty can be reduced by taking an ADD)	Impact on Confidentiality (Potential impact on confidentiality requirements): Direct (Direct impact on confidentiality)
Resolution Time (Time at which the uncertainty is expected to be fully resolved): Requirements Time (As soon as requirements are defined, the uncertainty is resolved)	Severity of Impact (Describes the severity if uncertainty is not mitigated): High (Total loss of confidentiality, or sensitive data)
Related to: How is communicated, Relation type: child	
Example: Within a mobility system, a company might offer Vehicle rentals. A user can rent a vehicle via a mobile app. The company then collects all necessary information and saves it in a database by sending the data to the database service. The database service uses an anonymization mechanism to anonymize the data transmitted by the company to hthem and then save it in the database. Upon requesting data, it uses the same mechanism to request the data from the database.	

Detail listing “Is the data anonymized?”

4 - How is communicated?	
Pertains to the uncertainty regarding the methods of communication employed by connectors within the archictecture.	
Keywords: Data	
Location (Describes where an uncertainty manifests itself within the architecture): System Behaviour (Behaviour of the system and its components as well as their communication)	Type (How much is known about the uncertainty and how it can be described): Scenario Uncertainty (Distinct scenarios depending on the uncertain outcome, no statistical means.)
AET (Elements to which an uncertainty can be assigned): Connector (Assignable to wires between components or communication)	Manageability (Can more knowledge or appropriate means reduce the uncertainty): Fully Reducible (Reducible)
Reducible by ADD (Uncertainty resolvable by an architectural design decision): Reducible (Uncertainty can be reduced by taking an ADD)	Impact on Confidentiality (Potential impact on confidentiality requirements): Indirect (Impact only in conjunction with contextual factors)
Resolution Time (Time at which the uncertainty is expected to be fully resolved): Design Time (As soon as the system is designed, the uncertainty is resolved)	Severity of Impact (Describes the severity if uncertainty is not mitigated): High (Total loss of confidentiality, or sensitive data)
Related to: Is the data anonymised?, Relation type: parent Is the communication encrypted?, Relation type: parent Is confidential data transmitted?, Relation type: parent	
Example: Within a mobility system, a company might offer Vehicle rentals. A user can rent a vehicle via a mobile app. The company then collects all necessary information and saves it in a database by sending the data to the database service. For example the data between Mobile App and Company might not be encrypted, thus disclosing personal information of the user.	

Detailed listing “How is communicated?”

Figure 5.1: Detailed listing of multiple uncertainties

5.3 Collaboration

As the knowledge on uncertainties is distributed among researchers and practitioners [26], the collection needs to be comprehensible and expandable for the community. To do so, a collaborative approach that allows for collecting knowledge across institutions and companies can be beneficial. Additionally, it allows for the maintenance of the collection, as the community can keep the knowledge up to date by refining, revising, and extending it. Only a few community members are needed to transfer the new knowledge gained from discussions into the collection. The collaborative approach also helps increase the quality of the knowledge due to the community being able to discuss the knowledge. In addition, it helps in creating a common understanding of uncertainties and sharing knowledge on mitigation strategies. Thus, supporting the explainability of uncertainties to strive to the fourth level of explainability [7], by enabling the exploration of explanations on a global level.

A collaborative approach should provide the ability to access an ongoing discussion on the knowledge without the need to pay for a license or subscription, as well as the ability to contribute to the discussion. Additionally, discussions on an uncertainty or classification category should be reachable from the uncertainty or classification category itself.

Consequently, the requirements of this approach from Section 4.2 need an extension with the following requirement for the collaborative approach:

R10: Enable Collaboration between software architects to capture and maintain knowledge on uncertainties without license or subscription constraints.

5.4 Realization

Chapter 4 introduced a set of requirements to tackle research questions **RQ1** and **RQ2**. Subsequently, Chapters 4 and 5 detailed an approach for fulfilling these requirements. So far, the approach addressed the requirements individually. Thus, combining them to address the research questions **RQ1** and **RQ2** falls upon the implementation phase. This section presents a realization that integrates these requirements.

To realize the requirements from Section 4.2 in a way that embeds **R10**, a web application can be helpful. It allows for reaching a broad audience and thus raises awareness on the topic of uncertainties regarding confidentiality. That is because it is reached through the Internet, making installations obsolete. Furthermore, licenses or subscriptions are not needed, and the location of the software architect is irrelevant. A web application can also use existing channels to enable communication and collaboration, which arguably other solutions cannot quickly provide. For example, it can use *GitHub*, a web-based platform that provides hosting for software development version control, facilitating collaboration and code sharing among teams to enable discussions and comments, as well as to incorporate new uncertainties. The web application can store all data and

information about uncertainties and their properties, as well as information about a chosen classification system.

The web application can employ a user interface, allowing easy information retrieval on uncertainties. It can also include different views to enable different ways of accessing the knowledge, fulfilling **R2**, **R3**, and **R4**. For example, using a table view to provide an overview of all uncertainties, as well as a detailed view of each uncertainty and classification category. Furthermore, the web application can provide the possibility to access the discussion on GitHub and allow cross-referencing between uncertainties and classification properties (**R9**). Additionally, the web application can provide the possibility to filter (**R7**) the uncertainties by different properties, as well as to search (**R8**) for a specific uncertainty. By arranging the data based on the meta-model proposed in Section 4.3, the web application can provide an overview of the relationships between uncertainties (**R5**) and applicable keywords (**R6**).

To show the feasibility of using a web application to implement the requirements, a prototype of the web application was implemented. It has three main views: The *Table* (Figure 5.3), the *Uncertainty Detail* (Figure 5.4) and the *Category Detail* (Figure 5.5) view.

At the beginning, the tool presents a welcome screen to the user (Figure 5.2), which states the purpose (**F1**). Clicking the button to see the table sends the user to the *Table* view (Figure 5.3).

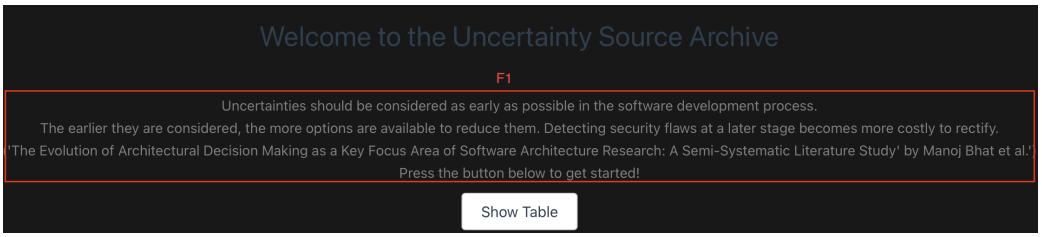


Figure 5.2: Welcome Screen, stating purpose of handling uncertainties (**F1**)

The *Table* view shows a set of 19 uncertainties in a table. It allows the user to filter (**F3**) the uncertainties by multiple properties, as well as search (**F4**) for a specific uncertainty. The user can reset all filters and the search with the reset button (**F5**). Additionally, the user can sort (**F2**) the table ascending and descending by clicking on the column headers. Clicking on an uncertainty redirects the user to the *Uncertainty Detail* view.

Here, the user has an overview of the definition (**F6**), the fitting keywords (**F7**), and the classification properties (**F8**), including a mouseover event to get additional descriptions on the classification category, as well as an on-click event, which redirects the user to the *Classification Detail* view. Additionally, the user can see all uncertainties related to the current uncertainty (**F10**), an example scenario, and an illustration of the scenario (**F9**). A button at the bottom of the page allows the user to get redirected to the discussion of the uncertainty on *GitHub* (**F11**).

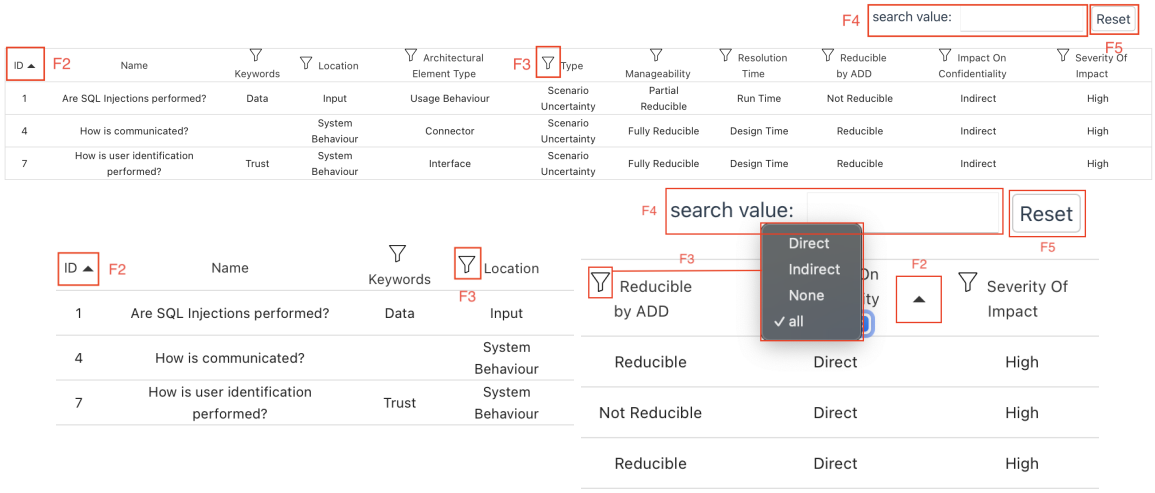


Figure 5.3: Table View of prototype including sorting (**F2**), filtering (**F3**), and searching (**F4**)

When a user clicks on a classification category, they get redirected to the *Classification Detail* view, where they can see all information about the classification category, including a description (**F12**), the possible options (**F13**), and a graphic representation of the classification category (**F14**). By clicking the buttons next to the options, the user gets sent back to the Table view with the filter set to the selected option (**F15**). Thus, the user gets a filtered view of all uncertainties with the selected option for the classification option. At the bottom of the page, the user can click on a button to get redirected to the discussion of the classification category on *GitHub* (**F16**). Table 5.1 provides an overview of all features.

5.5 Summary

This chapter addressed the information retrieval to conclude the Chapter 4 introduced approach to address **RQ1**, which asks to collect, manage, and make the knowledge of uncertainties regarding confidentiality comprehensible for software architects. Section 5.1 proposed using a user interface to increase the comprehensibility and retrieval of the information on uncertainties. Then, Section 5.3 introduced a collaborative approach to enable the extension and maintenance of the knowledge (**RQ2**). This approach allows for the collection of knowledge across institutions and companies, as well as the maintenance of the knowledge with manageable effort. Finally, Section 5.4 proposed to merge a web application with collaborative methods to facilitate the collection and management of knowledge concerning confidentiality-related uncertainties. The validation of the approach is still pending.

Welcome Screen:

F1: Stating Purpose

Table View:

F2: Sort Table Ascending/Descending

F3: Filter Table by Multiple Properties

F4: Search for specific Uncertainty

F5: Reset Filters and Search

Uncertainty Detail View:

F6: Show Definition

F7: Show Keywords

F8: Show Classification Properties

F9: Show Example

F10: Show Related Uncertainties

F11: Link to Discussion on GitHub

Classification Detail View:

F12: Show Description

F13: Show Options

F14: Show Graphic Representation

F15: Filter Table by Option

F16: Link to Discussion on GitHub

Table 5.1: Features included in prototype

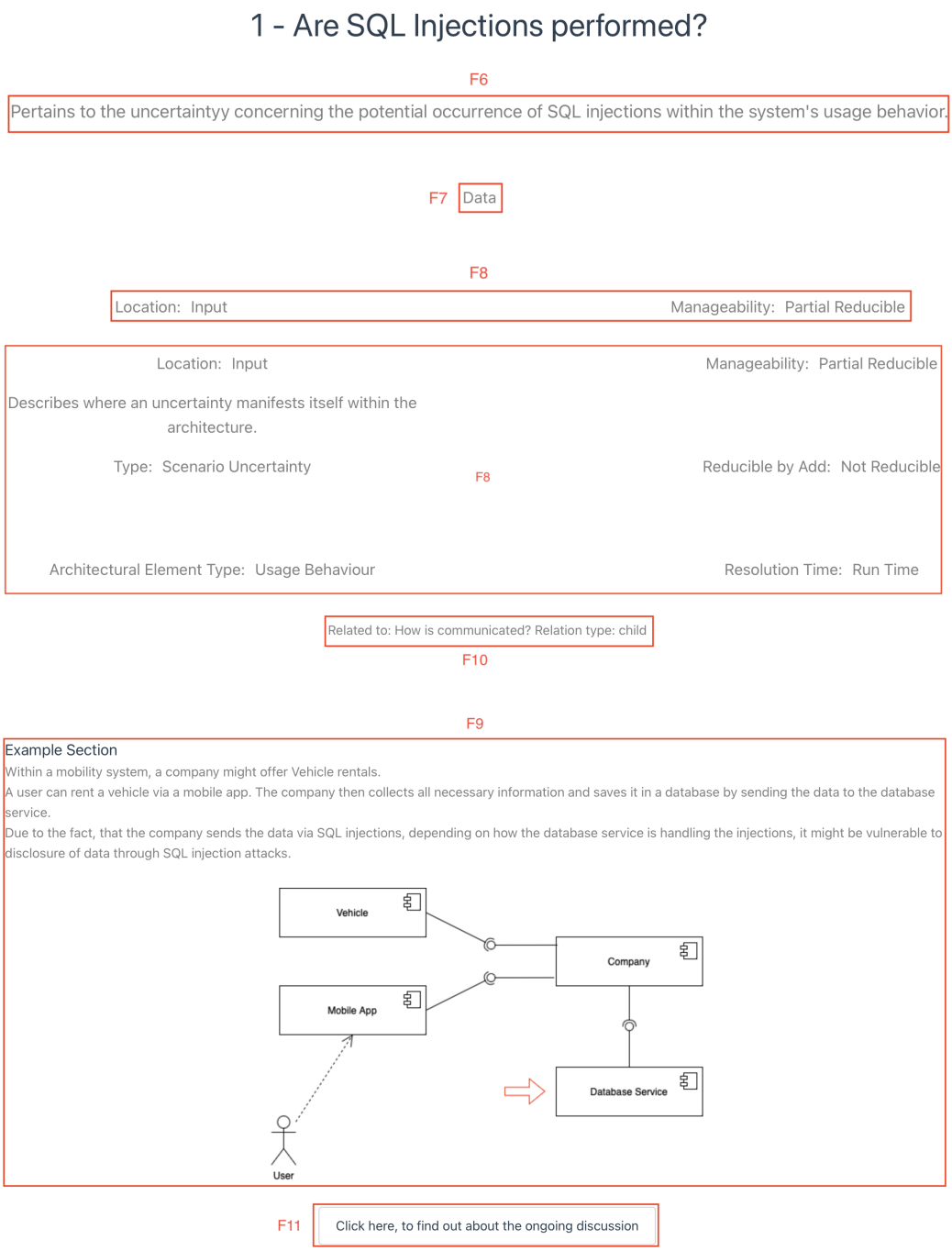


Figure 5.4: Uncertainty detail view of prototype including definition (**F6**), keywords (**F7**), classification properties (**F8**), example (**F9**), related uncertainties (**F10**) and link (**F11**)

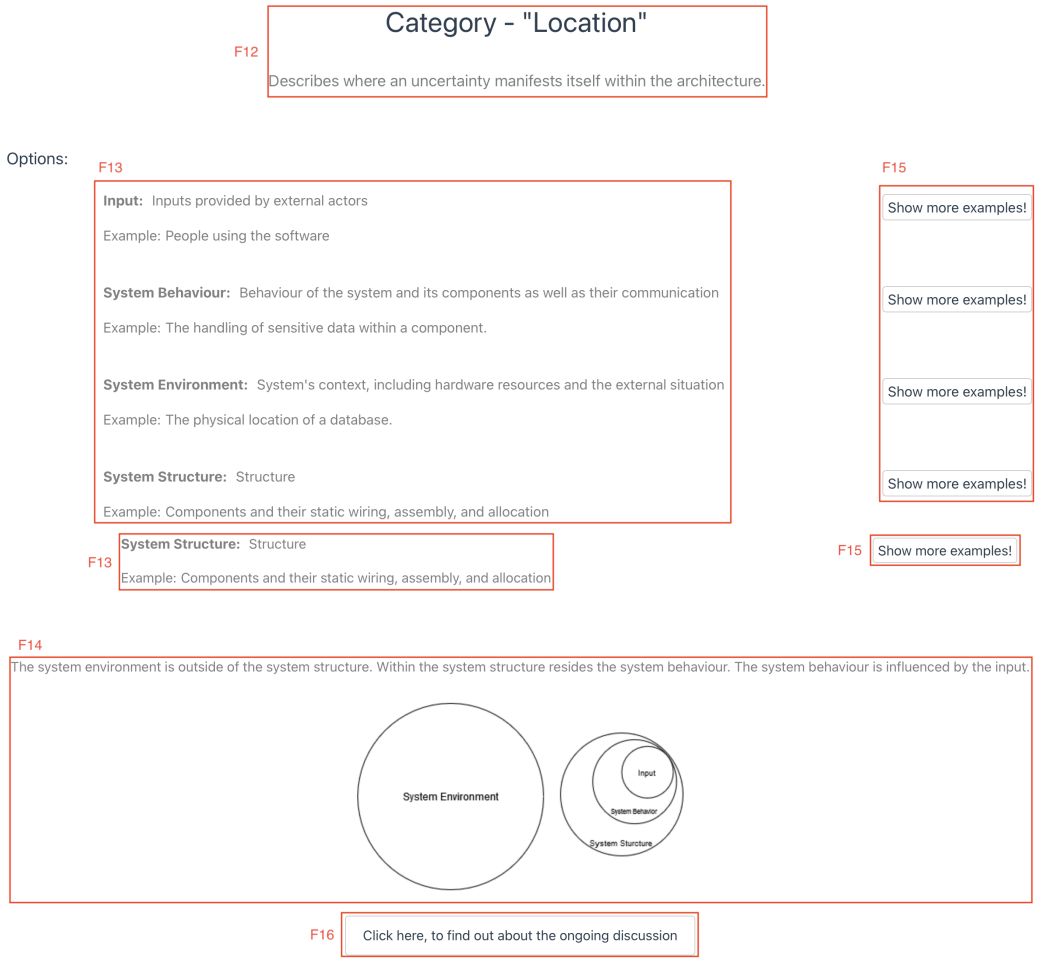


Figure 5.5: Category detail view of prototype, including description (**F12**), options (**F13**), graphic representation (**F14**), filtering for examples (**F15**), and link to discussion (**F16**)

6 Evaluation

This chapter outlines the evaluation of the proposed approach, conducted primarily based on the prototype introduced in Section 5.4. Section 6.1 presents the evaluation structure, employing a GQM-Plan, followed by the presentation of the evaluation design in Section 6.2. Subsequently, Section 6.3 presents and discusses the individual results. Additionally, Section 6.4 outlines the potential threats to validity, while Section 6.5 discusses the evaluation limitations.

6.1 Goals, Questions and Metrics

This evaluation uses a Goal-Question-Metric (GQM) approach [4] to ensure the quality of the results. The following gives an overview of the goals, questions, and metrics. Additionally, Table 6.1 shows the goals and questions in place.

G1: Providing a comprehensive tool for representing uncertainties that impact confidentiality based on the chosen classification [25].

- **Q1.1:** Are the requirements stated in the approach met by the features?
- **Q1.2:** Does the tool contain all classification categories?
- **Q1.3:** Can other observed uncertainties be described within the capabilities of the tool?
- **Q1.4:** Are any features unnecessary for meeting the requirements?
- **Q1.5:** Are any requirements unnecessary for the description of uncertainties?

G2: Providing an approach to assist software architects in making informed design decisions on uncertainties with data confidentiality implications in software designs.

- **Q2.1:** Can the tool assist software architects in identifying uncertainties in software designs?
- **Q2.2:** Does the tool enable software architects to collaborate with others?
- **Q2.3:** Does the tool provide a good user experience?

Table 6.1: Overview of goals and questions for evaluation

G1. This objective focuses on the representation qualities of the tool. It aims to assess whether the tool is consistent with the requirements stated in the approach and whether the requirements serve the purpose of comprehensively representing uncertainties affecting confidentiality.

Consequently, the first question, **Q1.1**, aims to confirm that the tool's features meet the requirements. For this purpose, **M1.1** measures the coverage of the requirements by the features. If the features cover all requirements, the coverage is 100%. Since the requirements include using a classification system, the second question, **Q1.2**, aims to validate if the tool includes all classification categories and options. To answer **Q1.2**, **M1.2** measures the ratio of classification categories incorporated in the tool compared to those not included. Since the approach aims to provide an extensible knowledge capture of uncertainties, the third question, **Q1.3**, aims to validate the tool's capabilities to describe uncertainties other than those observed during this research. For that, **M1.3** measures the proportion of uncertainties that the tool is capable of describing compared to the total amount of uncertainties attempted to describe. Questions **Q1.4** and **Q1.5** seek to confirm that the tool does not contain features that do not serve to meet the requirements, and no requirement is removable to simplify the approach without losing the ability to describe uncertainties. To evaluate on **Q1.4**, **M1.4** counts the number of removable features without compromising the capacity to fulfill the requirements. To assess **Q1.5**, **M1.5** calculates the percentage of uncertainties utilizing a requirement for description to the total number of uncertainties described in the prototype.

G2. This goal focuses on validating that the requirements can meet the intentions of the approach. It aims to validate that the requirements stated in the approach fulfill the intention to provide an approach to assist software architects in making informed design decisions on uncertainties, which may have implications on data confidentiality in software designs. Upon validating in goal **G1** that the prototype fulfills the requirements of the approach and does not include any additional features, it is deemed capable of presenting insights into the purpose of the approach via the prototype. Consequently, questions that aim to address **G1** include the usage of the prototype to provide insights into the approach.

The first question, **Q2.1**, aims to validate that the tool can assist software architects in identifying uncertainties in software designs. To evaluate the question, **M2.1** measures the proportion of uncertainties correctly identified to those not. The second question, **Q2.2**, assesses the collaboration capabilities proposed in Section 5.3. Its purpose is to confirm whether the tool facilitates collaboration among software architects. It is evaluated based on the percentage of uncertainties accurately identified and described through collaborative features (**M2.2**). The third question, **Q2.3**, focuses on evaluating the user experience of software architects to determine their inclination toward using the tool. The System Usability Score (SUS) (**M2.3**) addresses this question through the SUS questionnaire [10], a questionnaire commonly employed in diverse fields to evaluate and enhance overall user experience.

6.2 Evaluation Design

The evaluation comprises two stages. First, a *theoretical validation* evaluates the representation qualities of the tool (**G1**). Second, a *user study* assesses the tool's purpose (**G2**). The following gives a detailed overview of the design of this evaluation.

Theoretical Validation. Firstly, the prototype's features (Table 5.1) are mapped to the requirements stated in the approach (Section 4.2). That allows for validating the requirements (**Q1.1**) by measuring the coverage of the requirements by the features (**M1.1**). Secondly, to evaluate the containment of the classification (**Q1.2**), the classification categories stated in the approach of Hahner et al. [25] are compared to the ones in the prototype. A classification category counts as successfully incorporated (**M1.2**) if its name, description, and options are included and identical in the prototype. For the evaluation of the extensibility of the knowledge capture (**Q1.3**), efforts are made to incorporate uncertainties from the dataset of Hahner et al. [24] and Ramirez et al. [40] into the prototype. The uncertainties are then compared to the original uncertainties. If all aspects of the uncertainties are outlined in the prototype, the uncertainty is considered described (**M1.3**). Additionally, the aim is to ensure that the prototype does not contain any features that do not serve to meet the requirements (**Q1.4**). Sixteen subsets of features are constructed to achieve this, each missing a specific feature. This methodology facilitates the evaluation of whether a subset still fulfills the requirements. When such a subset is identified, the absent feature is classified as unnecessary (**M1.4**). Lastly, it is aimed to confirm that no requirement can be removed without losing the ability to describe uncertainties (**Q1.5**). This is evaluated by measuring the ratio of how many uncertainties use a requirement to how many uncertainties are described in the prototype (**M1.5**). Thus making a requirement most important, with a ratio of 1. Additionally, the results are compared to the amount of the requirements that were described in prior literature.

User Study. A user study involving 17 participants from the domain of software engineering was conducted. It is the author's opinion that given the objective of evaluating the supporting capabilities of the approach for software architects to make informed decisions (**RQ1**) and the extending capabilities of the approach (**RQ2**) through collaboration (**R10**), compared to interviews or a survey, a user study represents the most suitable choice. That is because using a user study enables the evaluation of the tool in a more realistic setting and provides insights into the tool's usage.

The user study contains three sections. Firstly, the user study asks participants to fill out a self-assessment in which they state their knowledge in different knowledge areas, such as *software architecture* and *uncertainties*. Additionally, it asks participants to choose the occupation that fits them best out of *student*, *researcher*, *practitioner*, and *others*. Secondly, the user study introduces participants to the research field of uncertainties, the classification used, and the tool. In this section, participants get familiar with the tool and its features. Subsequently, the user study presents two assignments that participants should complete within 30 minutes. The *first assignment* asks them to identify uncertainties in a given software design. They should find two fitting uncertainties in the tool. Only the first possesses an additional description. This assignment design allows first insights

on question **Q2.1**, which inquires about the tool's supporting capabilities for software architects in identifying uncertainties. As a subtask, participants should validate if a given uncertainty describes a given scenario and, if not, find a better-fitting uncertainty. Its design allows insights into question **Q2.2**, the collaboration capabilities of the tool, as the answer lies in the discussion of the given uncertainty on GitHub. This design also enables to measure **M2.2**, the amount of uncertainties that are correctly identified and described through collaboration, as participants explain their reasoning for their choice. The *second assignment* asks them to find at least two uncertainties in a given scenario. This assignment design simulates a scenario close to the real world, where software architects having a software architectural model and user stories, want to identify relevant uncertainties that need mitigation. Consequently, the expectation is to get the most insights on question **Q2.1**, the supporting capabilities of the tool for software architects in identifying uncertainties. To evaluate **Q2.1**, two people with good knowledge of uncertainties determine the correctness of the answers given by participants to measure **M2.1**. They evaluate the correctness based on a "gold standard", which results from a consensus on their independent assessment of correct uncertainties. After the assignments, the feedback section of the user study asks participants to fill out a System Usability Score (SUS) questionnaire [4] and to give feedback on their opinion on the usability and intuitiveness of the features on a scale from 1 strongly disagree to 4 strongly agree which allows measuring **M2.3**. Together with feedback on what participants think they learned through using the tool, this should give insights to answer **Q2.3**, the user experience.

6.3 Evaluation Results and Discussion

This section presents and discusses the evaluation results for each question individually. Beginning with the presentation of results for the theoretical validation in Section 6.3.1, the discussion proceeds to the findings of the user study in Section 6.3.2.

6.3.1 Theoretical Validation Results

Beginning with the outcomes of the feature-to-requirement mapping (**Q1.1**), Table 6.2 illustrates all features directly corresponding to a requirement. The findings indicate the presence of at least one feature for each requirement. However, the only requirement that remained unmapped was **R1**. That does not come as a surprise since mapping features to requirements alone does not provide insights into the comprehensibility of the classification. Fortunately, Hahner et al. [25] already evaluated the comprehensibility of their classification. With their evaluation of ease of use, they evaluated their classification with a System Usability Score (SUS) questionnaire and a SUS score of 68.25. With a score above average, it is assumable that the classification is comprehensible. Thus, being able to map at least one feature to all other requirements and with a coverage (**M1.1**) of 100%, the requirements are met by the features of the prototype.

Requirement	Feature	Requirement covered?
R1	**F4**, **F8**, **F12**, **F13**, **F14**	✓
R2	**F6**, **F8**, **F9**, **F10**	✓
R3	**F12**, **F13**, **F14**, **F15**, **F16**	✓
R4	**F9**, **F11**	✓
R5	**F10**	✓
R6	**F7**	✓
R7	**F2**, **F3**	✓
R8	**F4**	✓
R9	**F7**, **F10**, **F15**	✓
R10	**F11**, **F16**	✓

Table 6.2: Feature-Requirement mapping and presence analysis

The next objective is to validate the inclusion of all classification categories within the prototype, addressing **Q1.2**. Table 6.3 shows the successful integration of all classification categories into the prototype, yielding a 100% incorporation rate when calculating the ratio between the number of classification categories incorporated in the tool (**M1.2**). This outcome was anticipated, given that the prototype's structure incorporates the classification system.

Classification Category from [25]	Classification Category in Prototype	Identical
Location	Location	✓
Architectural Element Type	Architectural Element Type	✓
Type	Type	✓
Manageability	Manageability	✓
Resolution Time	Resolution Time	✓
Reducible by ADD	Reducible by ADD	✓
Impact on Confidentiality	Impact on Confidentiality	✓
Severity of Impact	Severity of Impact	✓

Table 6.3: Classification categories included in prototype

For the evaluation of the knowledge capture's extensibility (**Q1.3**), the uncertainties of *Missing Requirements*, *Unexplored Alternatives*, and *Incomplete Information* from Ramirez et al. [40], cited as examples in their paper, along with uncertainties from the dataset of Hahner et al. [24], were chosen to be integrated into the prototype. Table 6.4 shows the results. The main challenge when trying to incorporate the uncertainties from Ramirez et al. [40] was that the uncertainties are not limited to the impact on confidentiality. Thus, being unable to describe the *Impact on Confidentiality* and *Severity of Impact* categories. Additionally, with limited knowledge of their uncertainties, it was not possible to adequately fill in the property for *Reducible by ADD* without investigating the uncertainties further. That did not influence the ability to transfer the information from the uncertainties of Ramirez et al. [40] into the prototype. Consequently, it was not deemed relevant to investigate the missing attributes further.

Since the prototype uses the classification system of [25] and **Q1.2** verified the use of the classification system, extending the knowledge with uncertainties from [24] was straightforward. Table 6.5 shows an example. Regarding the understanding of "described", which Section 6.2 specifies, the results show that all aspects of the chosen uncertainties are describable under the usage of **M1.3**. Consequently, the prototype can describe other uncertainties than the ones observed by Hahner et al. [25], thus addressing question **Q1.3**.

Sixteen subsets of features were constructed, each containing all but one feature, to evaluate the necessity of the features (**Q1.4**). It was tried to identify subsets that still meet the requirements, which would make them redundant using **M1.4**. The results in Table 6.6 show against which requirements the subsets failed. It shows that features **F4**, the custom search feature, and **F5**, the feature to reset filters, and search values, are not failing, thus making them initially redundant. However, it implies they are still advantageous features, especially for the tool's usability. Otherwise, one would be unable to find uncertainties based on a single thought or reset filters without effort. Therefore, the results suggest that all features are essential for the functionality and effectiveness of the tool.

The final evaluation of the *theoretical validation* addresses **Q1.5**, which asks for the necessity of each requirement. To answer **Q1.5**, **M1.5** measures the ratio of how many uncertainties included in the prototype use a requirement. The results in Table 6.7 show that most uncertainties use all requirements. Exceptions are the hierarchical structures (**R5**) with a requirement usage ratio of 57.9% and keywords (**R6**), with a requirement usage ratio of 68.5%. As the requirement ratio of **R5** and **R6** is still above 50% and combined with the common usage of the requirements in existing literature in other contexts, the results suggest that the requirements have their right to be used to describe uncertainties. Therefore, the results imply that all requirements are necessary to describe uncertainties.

The results of the *theoretical validation* confirm that the features meet the requirements (**Q1.1**). They show that the tool contains all classification categories (**Q1.2**) and can be used to describe other uncertainties than the ones observed (**Q1.3**). Additionally, based on the results, all features are necessary for the functionality and effectiveness of the tool (**Q1.4**), and all requirements are necessary to describe uncertainties (**Q1.5**). Consequently, the theoretical validation implies that the tool is capable of representing uncertainties with an impact on confidentiality comprehensively (**G1**).

Property	Value	Extracted from original Property
Uncertainty:	Have all requirements been stated?	Name
Keyword:	Requirements	Classification
Resolution Time:	Requirements Time	Context
Location:	System Behavior	Related Resources
Architectural Element Type:	Interface/Connector	Impact
Type:	Scenario Uncertainty	Context
Impact on Confidentiality:	Indirect	Impact
Manageability:	Fully Reducible	Context
Reducible by ADD:		
Severity of Impact:		
Example:	Illustration Example	Sample Illustration
In Discussion:	Mitigation Strategies	Mitigation Strategies
Relationship:	Incomplete Requirements	Also Known as

(a) Uncertainty “Missing Requirements” from [40]

Property	Value	Extracted from original Property
Uncertainty:	Have all alternatives been explored?	Name
Keyword:	Alternatives, Design	Classification
Resolution Time:	Design Time	Context
Location:	System Structure	Impact/Context
Architectural Element Type:	Component/Connector	Related Resources
Type:	Recognized Ignorance	Context/Impact
Impact on Confidentiality:	Indirect	Impact
Manageability:	Fully Reducible	Context
Reducible by ADD:		
Severity of Impact:		
Example:	Illustration Example	Illustration Sample
In Discussion:	Mitigation Strategies	Mitigation Strategies
Relationship:	Parent of: How is communicated? (U4), Which component is chosen? (U39)	

(b) Uncertainty “Unexplored Alternatives” from [40]

Property	Value	Extracted from original Property
Uncertainty:	Is all information available?	Name
Keyword:	Data	Context
Resolution Time:	Run Time	Classification
Location:	System Environment	Impact
Architectural Element Type:	Connector	Impact
Type:	Scenario Uncertainty	Impact
Impact on Confidentiality:	Indirect	Impact
Manageability:	Partial Reducible	Mitigation Strategies, Context, Impact
Reducible by ADD:		
Severity of Impact:		
Example:	Illustration Example	Sample Illustration
In Discussion:	Mitigation Strategies	Mitigation Strategies
Relationship:	Parent of: Is the data entered correct? (U17), What data is entered? (U24)	

(c) Uncertainty “Incomplete Information” from [40]

Table 6.4: Uncertainty described by Ramirez et al. [40] transferred to tool

Property	Value	Extracted from original Property
Uncertainty:	Are SQL Injections performed?	Uncertainty
Keyword:	Data	Uncertainty
Resolution Time:	Run Time	Resolution Time
Location:	Input	Location
Architectural Element Type:	Usage Behavior	Architectural Element Type
Type:	Scenario Uncertainty	Type
Impact on Confidentiality:	Indirect	Impact on Confidentiality
Manageability:	Partial Reducible	Manageability
Reducible by ADD:	No	Reducibly by Add
Severity of Impact:	High	Severity of Impact
Example:		
In Discussion:		
Relationship:	Child of: How is communicated? (U4)	

Table 6.5: Uncertainty "Are SQL Injections performed" from [24] transferred to tool

Subset ID	Missing Feature	Failing against	Reason
1	**F1**		
2	**F2**	**R7**	No sorting mechanisms
3	**F3**	**R7**	No filtering mechanisms
4	**F4**	**R8**	No search mechanisms
5	**F5**		
6	**F6**	**R2**	Definition Mandatory in Comprehension of Uncertainty
7	**F7**	**R6**	No keyword categorization possible
8	**F8**	**R1**	Not using any classification system for structuring Uncertainties
9	**F9**	**R4**	Example Scenario plays important role in giving context
10	**F10**	**R5**	Not supporting any relationships structures between uncertainties
11	**F11**	**R10**	Not supporting collaboration mechanisms
12	**F12**	**R3**	Category description essential for classification terminology comprehension
13	**F13**	**R3**	Category options and their description essential for classification terminology comprehension
14	**F14**	**R4**	Another dimension of classification terminology dimension missing
15	**F15**	**R4**	Missing examples that give context to category option
16	**F16**	**R10**	Not supporting collaboration mechanisms

Table 6.6: Subsets of features and if they still meet the requirements

Requirement	Used Amount	Used %	Used in Prior Work
R1	19	1	[30]
R2	19	1	[13], [30], [20], [2], [21]
R3	8	1	
R4	19	1	[13], [2], [30]
R5	11	0.579	[13], [2], [54], [30]
R6	13	0.685	[13], [2], [30]
R7	19	1	[20]
R8	19	1	[20], [2], [30]
R9	19	1	
R10	27	1	[13], [20], [2], [15], [21], [54], [30]

Table 6.7: Requirements used by uncertainties and in existing literature

6.3.2 User Study Results

During the user study, data was collected from 17 participants, seven of whom were students, five researchers, and five practitioners (Figure 6.1). None of the participants had to choose "other" as their occupation.

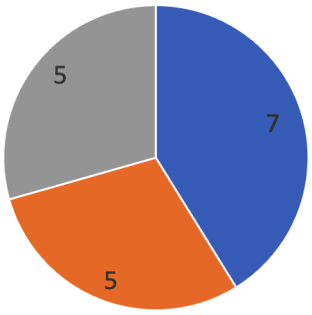


Figure 6.1: Participants Composition: blue - Student, orange - Researcher, grey - Practitioner, green - Other

The participants' self-assessment in Table 6.8 indicates that the average participant had good knowledge of *software architecture* and *architectural design decisions*, little knowledge of *confidentiality*, *uncertainty*, *risk assessment* and *security analysis*, and no knowledge of *knowledge management systems*. It also shows that almost none of the participants described themselves as experts in any of the knowledge areas, except for *architectural design decisions*, with 29.4% of the participants. Regarding the essential knowledge areas for this research, *software architecture*, *confidentiality*, and *uncertainties*, more than half of the participants had good knowledge of *software architecture* and little knowledge in *confidentiality*, and *uncertainties*.

Knowledge Area	No Knowledge	Little Knowledge	Good Knowledge	Expert
Software Architecture	0	0.176	0.706	0.118
Architectural Design Decisions	0	0.294	0.412	0.294
Confidentiality	0.235	0.588	0.059	0.118
Uncertainty	0.294	0.412	0.235	0.059
Risk Assessment	0.294	0.353	0.353	0
Knowledge Management Systems	0.353	0.294	0.353	0
Security Analysis	0.353	0.471	0.118	0.059

Table 6.8: Amount of knowledge participants have of different knowledge areas, based on their self-assessment.

To answer **Q2.1**, **M2.1** measured the accuracy of participants in describing uncertainties correctly. In Assignment 1, 79.4% of all answers described a correctly identified uncertainty,

as shown in Table 6.9. 94,1% of all participants described at least one correct uncertainty (Table 6.10). The results of Assignment 1 show that participants described Uncertainty One correctly more often than Uncertainty Two. That is likely because, for Uncertainty One, participants received an additional description elucidating the underlying uncertainty. In contrast, for Uncertainty Two, they were tasked with identifying the uncertainty solely based on the scenario and the software architectural model, with an indication of the uncertainty's position. Examining the correctness concerning the participants' occupations reveals that the researchers accurately outlined all uncertainties, whereas the students encountered the most challenges, exhibiting an average accuracy of 71.4%. When taking into account the prior knowledge of the participants, the results show that participants with no prior knowledge in the domain of *uncertainties* were the ones with the lowest correctness of 50%. Given the predictable nature of this outcome, due to the missing knowledge of the subject, the correctness of 92.9% among participants possessing limited expertise in the domain of *uncertainties* is particularly surprising. The suggested explanation is that these participants effectively utilized their limited knowledge to engage with the prototype, leading to a better understanding of the domain of *uncertainties* and thereby enabling accurate articulation of the uncertainties. They performed better than the participants with good knowledge in the domain of uncertainties, who had a correctness score of 87.5%. These results might suggest that their relatively poorer performance than participants with limited knowledge stems from their higher confidence in their understanding, resulting in underutilization of the prototype's full potential.

Category		Uncertainty One		Uncertainty Two		Total	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
	Overall	0.824	0.176	0.764	0.235	0.794	0.206
Profession	Student	0.714	0.286	0.714	0.286	0.714	0.286
	Researcher	1	0	1	0	1	0
	Practitioner	0.8	0.2	0.8	0.2	0.8	0.2
Prior Uncertainty Knowledge	No	0.4	0.6	0.6	0.4	0.5	0.5
	Little	1	0	0.857	0.143	0.929	0.071
	Good	1	0	0.75	0.25	0.875	0.125
	Expert	1	0	1	0	1	0

Table 6.9: Correctness of uncertainty descriptions in % in Assignment 1, based on the "gold standard"

The design of Assignment 2 is the closest to the reality use case of the prototype. Exhibiting an overall ratio of accurately identified uncertainties (**M2.1**) at 95.3%. Table 6.11 presents these results. Alongside with 100% of participants accurately describing at least two uncertainties (6.12), the findings surpass initial expectations. Participants described uncertainties correctly, even though they were only given a scenario and an unedited software architectural model with no additional description. Examining the correctness concerning participants' occupations, it becomes evident that, similarly to Assignment 1, the researchers accurately described all uncertainties. Even students surpassed practitioners, attaining an average of 94.4% correct answers compared to practitioners' average of

Category		All Wrong	One Correct	Two Correct	≥ One Correct
	Overall	0.059	0.294	0.647	0.941
Profession	Student	0.143	0.286	0.571	0.857
	Researcher	0	0	1	1
	Practitioner	0	0.6	0.4	1
Prior Uncertainty Knowledge	No	0.2	0.6	0.2	0.8
	Little	0	0.143	0.857	1
	Good	0	0.25	0.75	1
	Expert	0	0	1	1

Table 6.10: Distribution of correctness across participants in % in Assignment 1

92.9%. When taking into account the prior knowledge of the participants, results show that participants with no prior knowledge in the domain of *uncertainties* were the ones with the lowest correctness of 92.3%. As the lowest performance is not a surprise, being able to describe uncertainties correctly with a score of 92.3% without any prior knowledge of the domain of *uncertainties* is unexpected. Thus, considering the results of Assignment 1 and Assignment 2, they imply that the prototype can support software architects in identifying uncertainties, which addresses **Q2.1**.

Category		Correct	Incorrect
	Overall	0.953	0.047
Profession	Student	0.944	0.556
	Researcher	1.000	0.000
	Practitioner	0.929	0.071
Prior Uncertainty Knowledge	No	0.923	0.077
	Little	0.944	0.0556
	Good	1.000	0.000
	Expert	1.000	0.000

Table 6.11: Correctness of uncertainty descriptions in % in Assignment 2, based on the "gold standard"

The design of Assignment 1.1 aims to estimate the collaboration capabilities of the prototype (**Q2.2**). The results show that none of the explanations included the correct answer, which could be found in the discussion section 6.13. When using **M2.2** to measure the collaboration capabilities of the prototype, the percentage of participants who correctly described the uncertainty using collaboration features is 0%. That nobody used the collaboration features to describe the uncertainty correctly might be because this user study was not conducted in a collaborative environment. Thus, participants might not have considered using the GitHub issue to find the correct answer. Furthermore, the wrong answer stated in the user study was too close to the correct answer, thus making it hard for the participants to identify the correct answer without the help of the discussion. The

Category		All Wrong	One Correct	Two Correct	Three Correct	≥ Two Correct
	Overall	0	0	0.588	0.412	1
Profession	Student	0	0	0.571	0.429	1
	Researcher	0	0	0.8	0.2	1
	Practitioner	0	0	0.4	0.6	1
Prior Uncertainty Knowledge	No	0	0	0.6	0.4	1
	Little	0	0	0.571	0.429	1
	Good	0	0	0.75	0.25	1
	Expert	0	0	0	1	1

Table 6.12: Distribution of Correctness across Participants in % in Assignment 2

initial idea behind this choice was that users would be inclined to use collaboration support over information stated in the prototype when they cannot decide between two answers that are very close to each other. Hence, the hints to utilize the discussion for answering Assignment 1.1 were insufficient, as there was no indication that the participants had even read the discussion entry. Nevertheless, 64.7% of the participants described the uncertainty correctly without using the collaboration features. The results concerning the participants' occupations show that practitioners struggled to describe the uncertainties correctly. As people with no knowledge described the uncertainty correctly in more cases than people with little knowledge of uncertainties, it underlines that the user study design for Assignment 1.1 has been flawed. Consequently, the results imply that the user study design needs to be optimized to evaluate the collaboration capabilities of the prototype. It can be summarized that in the prototype's evaluation, the collaboration capabilities cannot be claimed, leaving **Q2.2**, which asks about the collaboration capabilities of the approach, unanswered.

Category		Correct	Incorrect	Correct using Collaboration
	Overall	0.647	0.353	0
Profession	Student	0.857	0.143	0
	Researcher	0.8	0.2	0
	Practitioner	0.2	0.8	0
Prior Uncertainty Knowledge	No	0.6	0.4	0
	Little	0.429	0.571	0
	Good	1	0	0
	Expert	1	0	0

Table 6.13: Correctness of uncertainty descriptions in % in Assignment 1.1, based on the "gold standard", and the use of collaboration features

To evaluate the user experience of the participants (**Q2.3**), they filled out the System Usability Scale (SUS) questionnaire (**M2.3**). Additionally, participants gave feedback on their opinion on the intuitiveness and usefulness of the different features on a scale from 1 to 4. With a SUS score of 69.7 [19], the prototype is above the average SUS score of 68.0 and

thus can be considered usable. Table 6.14 shows the average intuitiveness and usefulness of the features. The results suggest that filtering (**F3**), examples (**F9**), and visualizations (**F14**) as features are most helpful. That implies that additional dimensions of support, such as graphical representations of uncertainties, are beneficial for the user experience. On the other hand, while the participants assessed the usefulness of most features as valuable, the hierarchical relationships feature (**F10**) was assessed as less valuable. A reason for that comes from the limited amount of uncertainties in the prototype, making it impractical to utilize the feature entirely. Additionally, as the participants assessed this feature as less intuitive, it implies that the implementation of this feature in the prototype was not intuitive enough, which impacts the usefulness of the feature. The additional participant feedback supports these results, as participants often mentioned the need for a better UI/UX experience [19]. Thus, addressing **Q2.3**, results suggest that the prototype would benefit from a better UI/UX experience to increase the intuitiveness and usefulness of the features. Nevertheless, the results show that participants assessed most of the prototype's features as useful and intuitive by the participants.

Participants also gave their opinions on what they learned about different topics during the user study. They evaluated the topics *software architecture*, *confidentiality*, *uncertainties and their impact*, *different types of uncertainties*, *benefits of collaboration*, and *benefits of the tool vs. a list* on a scale from 1 to 4, with one strongly disagreeing and four strongly agreeing. Table 6.15 shows the average opinion of the participants. The results show that participants learned the most about *different types of uncertainties* and *uncertainties and their impact*, with an average value of 3.69 and 3.38, respectively. Additionally, they learned about *confidentiality* and *benefits of the tool vs. list*, with an average value of 3.08. As the participants could not use the collaboration features properly, as stated in prior results, it is not surprising that they did not learn about the *benefits of collaboration*, with an average value of 1.77. The results support the research goal **G2**, as the participants learned about the different uncertainties and their impact.

Feature	Usefulness	Intuitiveness
F3	3.563	3.235
F4	3.214	3.588
F7	3.2	3.5
F10	2.333	2.067
F9	3.765	3.176
F14	3.765	3.375
F11, **F16**	3	3.25
F8 (OnClick)	2.867	2.667
F8 (Mouseover)	3	2.313

Table 6.14: Average intuitiveness and usefulness of features. Scale: 1 - Strongly Disagree, 2 - Disagree, 3 - Agree, 4 - Strongly Agree

Consequently, the results of the user study suggest that the prototype can assist software architects in identifying uncertainties in software designs (**Q2.1**), and it provides a reason-

able user experience (**Q2.3**). However, it would benefit from a better UI/UX experience. Unfortunately, the results were inconclusive regarding the collaboration capabilities of the prototype (**Q2.2**). Subsequently, the results suggest that the approach can assist software architects in making informed design decisions on uncertainties with data confidentiality implications in software designs (**G2**), with the validation of the collaboration capabilities being left for future work.

I learned about…	Average Value
Software Architecture	2.85
Confidentiality	3.08
Uncertainties and their Impact	3.38
Different Types of Uncertainty	3.69
Benefits of Collaboration	1.77
Benefits of Tool vs List	3.08

Table 6.15: Average opinion on the value of this approach by participants. Scale: 1 - Strongly Disagree, 2 - Disagree, 3 - Agree, 4 - Strongly Agree

6.4 Threats to Validity

The threats to validity are discussed based on the guidelines of Runeson et al. [42]. The most significant threat identified regarding *internal validity* is the assessment of the assignment results of the user study by two people with good knowledge in the context of this research. Although it was endeavored to minimize the subjectivity of the assessment by defining a clear set of rules and assessing the results independently, the results could have been different if other people had assessed the assignments. Additionally, the user study was conducted in 9 online meetings, which could have led to different information given to the participants and thus led to different results. A clear structure of the meetings and the information given to the participants attempted to minimize this threat. However, there is a chance that participants received different information, especially when they had questions before they started the assignments. Regarding *external validity*, the most significant threat identified is the number of participants. Despite attempts to diversify the participants based on occupation, 17 participants do not provide a representative sample of all software architects. Nonetheless, the results still give a good indication of the usefulness of the approach, as the participants described the uncertainties correctly throughout their diversified occupations and prior knowledge. Another threat to *external validity* is the chosen set of uncertainties included in the prototype and used in the user study. Although the set of uncertainties was chosen based on the literature, other uncertainties could have led to different results. A GQM-based evaluation plan was applied (Section 6.1) to limit the threat of *construct validity*. For the *reliability* and *replicability* of the results, the prototype and user study are published [19].

6.5 Limitations

The most significant limitation identified for this approach is using a single classification system, **R1**. Thus, the limitations of the chosen classification system are also limitations of this approach. In this case, the limitations of the classification system described by Hahner et al. [25], such as the "focus on confidentiality as central quality attribute" [25], are also limitations of this approach. Additionally, this approach does not support downloading or exporting the identified uncertainties directly into analysis or mitigation approaches. That is a limitation, as it requires the user to manually transfer the identified uncertainties into the analysis or mitigation approaches, which is time-consuming and error-prone. As other limitations may exist, the extensibility capabilities of this approach might allow the possibility to address them with manageable effort in future work.

7 Conclusion

This chapter summarizes the research and contributions this thesis offers in Section 7.1. Finally, Section 7.2 provides an overview of future work derived from the research findings.

7.1 Summary

This thesis addressed the missing awareness of uncertainties regarding confidentiality across software architects and the need for an overview of current knowledge on uncertainties regarding confidentiality. It aimed to contribute to the collection, management, and improvement of comprehensibility of the knowledge on uncertainties regarding confidentiality (**RQ1**). Furthermore, it aimed to enable extensibility and maintainability of the uncertainty collection through collaboration (**RQ2**). Firstly, this included the engineering of requirements, which resulted from observations and findings in existing literature, for an approach that allows for the collection and management of the knowledge on uncertainties regarding confidentiality (Section 4.2). These requirements included the use of a classification system (**R1**), adding context to uncertainties like an example or visual representation (**R4**), and the possibility of relating uncertainties to each other (**R5**). The approach used the requirements to develop solutions to address the research questions. That included a meta-model that allows for the description and relation of and between uncertainties regarding confidentiality (Section 4.3). Furthermore, the meta-model can describe uncertainties that might be identified in future research (Section 5.2). The initial approach was adjusted to maintain and extend the collection of uncertainties. That adjustment enabled the collaboration between researchers and practitioners (Section 5.3). These collaboration mechanisms enable discussions about uncertainties regarding confidentiality and new findings, as well as enable the collection of knowledge across institutions and companies. Then, this thesis presented the realization of the approach as a web application in the form of a prototype (Section 5.4). The prototype served as the base to evaluate the approach. The evaluation was funded on a Goals-Question-Metrics Plan (GQM) (Section 6.1) and resulted in an evaluation design based on a theoretical evaluation and a user study (Section 6.2). The theoretical evaluation assessed the approach's ability to fulfill **RQ1**. It showed that the approach allows the description and relation of uncertainties and can improve the comprehensibility of the knowledge on uncertainties. The user study evaluated the approach's ability to fulfill **RQ2** and was conducted with 17 people from the field of software engineering, divisible into seven students, five researchers, and five practitioners. The results showed that in a close-to real-world scenario, the approach

could support software architects to identify uncertainties regarding confidentiality, as the participants identified and described uncertainties correctly with an accuracy of 95.3%. The results also showed that little knowledge about uncertainties is enough to use the approach effectively, as the participants with little prior knowledge identified and described uncertainties with an accuracy of 92.9% in Assignment 1 compared to 50% accuracy on participants with no knowledge of uncertainties. However, the results also show that the approach's collaborative qualities require further investigation, as the results were inconclusive on the approach's ability to enable collaboration between researchers and practitioners due to the user study's design. Consequently, the proposed evaluation could not fully evaluate **RQ2** regarding the extendability and maintainability of the uncertainty collection. Thus, this thesis could not provide a funded answer for **RQ2**.

This thesis contributes to the ongoing efforts to improve the confidentiality of software systems by providing an approach that allows for the collection and management of knowledge on uncertainties regarding confidentiality. Software architects can use the collection of Uncertainties to identify and understand uncertainties regarding confidentiality and use it to improve analysis methods and develop mitigation strategies. Furthermore, the approach can raise awareness and increase the understanding of uncertainties regarding confidentiality across software architects.

7.2 Future Work

The evaluation results indicated that there is still progress to be made in addressing **RQ2**. Collaboration between researchers and practitioners remains a crucial aspect of this approach. The evaluation design employed needs to be revised in assessing the collaborative qualities of the approach, leaving it to future work to evaluate its ability to enable collaboration qualities for the collection of knowledge on uncertainties regarding confidentiality. As the approach prototype insinuated the possibility of collaboration and adding new uncertainties to the uncertainty collection, this is a potential foundation for future work in this domain. Integrating bots for validating new uncertainties and thus potentially automating the process of adding new uncertainties could be a valuable addition to managing the effort to maintain the uncertainty collection.

Moreover, the existing research on the correlations among uncertainties concerning confidentiality is limited. The proposed approach could facilitate the collection of additional uncertainties, consequently fostering further research on the interconnections between uncertainties. Furthermore, by expanding the pool of uncertainties within the collection, this approach could serve as a valuable foundation for investigating the prioritization of confidentiality-related uncertainties and the potential trade-offs between quality attributes.

Bibliography

[1] Maribel Acosta et al. “Uncertainty in coupled models of cyber-physical systems”. In: *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. MODELS ’22. New York, NY, USA: Association for Computing Machinery, Nov. 9, 2022, pp. 569–578. ISBN: 978-1-4503-9467-3. DOI: 10.1145/3550356.3561539. URL: https://dl.acm.org/doi/10.1145/3550356.3561539 (visited on 06/08/2023).

[2] Muhammad Ali Babar, Xiaowen Wang, and Ian Gorton. “PAKME: A Tool for Capturing and Using Architecture Design Knowledge”. In: *2005 Pakistan Section Multitopic Conference*. 2005 Pakistan Section Multitopic Conference. Dec. 2005, pp. 1–6. DOI: 10.1109/INMIC.2005.334419.

[3] Jae Young Bang et al. “How software architects collaborate: Insights from collaborative software design in practice”. In: *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). San Francisco, CA, USA: IEEE, May 2013, pp. 41–48. ISBN: 978-1-4673-6290-0. DOI: 10.1109/CHASE.2013.6614730. URL: http://ieeexplore.ieee.org/document/6614730/ (visited on 10/21/2023).

[4] Victor R. Basili and David M. Weiss. “A Methodology for Collecting Valid Software Engineering Data”. In: *IEEE Transactions on Software Engineering* SE-10.6 (Nov. 1984). Conference Name: IEEE Transactions on Software Engineering, pp. 728–738. ISSN: 1939-3520. DOI: 10.1109/TSE.1984.5010301.

[5] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Google-Books-ID: mdiIu8Kk1WMC. Addison-Wesley Professional, 2003. 572 pp. ISBN: 978-0-321-15495-8.

[6] Niko Benkler. “Architecture-based Uncertainty Impact Analysis for Confidentiality”. Master’s thesis. Karlsruher Institut für Technologie, Feb. 23, 2022.

[7] Marcello M. Bersani et al. “A Conceptual Framework for Explainability Requirements in Software-Intensive Systems”. In: *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*. 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW). Hannover, Germany: IEEE, Sept. 2023, pp. 309–315. ISBN: 9798350326918. DOI: 10.1109/REW57809.2023.00059. URL: https://ieeexplore.ieee.org/document/10260808/ (visited on 11/01/2023).

[8] Manoj Bhat et al. “The Evolution of Architectural Decision Making as a Key Focus Area of Software Architecture Research: A Semi-Systematic Literature Study”. In: *2020 IEEE International Conference on Software Architecture (ICSA)*. 2020 IEEE International Conference on Software Architecture (ICSA). Mar. 2020, pp. 69–80. DOI: 10.1109/ICSA47634.2020.00015.

[9] Barry Boehm. “Software risk management”. In: *ESEC ’89: 2nd European Software Engineering Conference University of Warwick, Coventry, UK September 11-15, 1989 Proceedings* (1989). Ed. by C. Ghezzi and J. A. McDermid, pp. 1–19.

[10] John Brooke. “SUS - A quick and dirty usability scale”. In: (1996), pp. 189–194.

[11] Bundeskriminalamt. *Cybercrime Bundeslagebild*. May 9, 2021. URL: https://www.bka.de/SharedDocs/Downloads/DE/Publikationen/JahresberichteUndLagebilder/Cybercrime/cybercrimeBundeslagebild2021.html?nn=28110 (visited on 05/30/2023).

[12] Hyun-Ju Choi et al. “Communities of practice and knowledge management systems: effects on knowledge management activities and innovation performance”. In: *Knowledge Management Research & Practice* 18.1 (Jan. 2, 2020), pp. 53–68. ISSN: 1477-8238, 1477-8246. DOI: 10.1080/14778238.2019.1598578. URL: https://www.tandfonline.com/doi/full/10.1080/14778238.2019.1598578 (visited on 10/21/2023).

[13] Michael Colesky and Julio C. Caiza. “A System of Privacy Patterns for Informing Users: Creating a Pattern System”. In: *Proceedings of the 23rd European Conference on Pattern Languages of Programs*. EuroPLoP ’18. New York, NY, USA: Association for Computing Machinery, July 4, 2018, pp. 1–11. ISBN: 978-1-4503-6387-7. DOI: 10.1145/3282308.3282325. URL: https://dl.acm.org/doi/10.1145/3282308.3282325 (visited on 08/15/2023).

[14] Michael Colesky et al. “A system of privacy patterns for user control”. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. SAC ’18. New York, NY, USA: Association for Computing Machinery, Apr. 9, 2018, pp. 1150–1156. ISBN: 978-1-4503-5191-1. DOI: 10.1145/3167132.3167257. URL: https://dl.acm.org/doi/10.1145/3167132.3167257 (visited on 08/15/2023).

[15] Naeem Esfahani, Sam Malek, and Kaveh Razavi. “GuideArch: Guiding the exploration of architectural solution space under uncertainty”. In: *2013 35th International Conference on Software Engineering (ICSE)*. 2013 35th International Conference on Software Engineering (ICSE). ISSN: 1558-1225. May 2013, pp. 43–52. DOI: 10.1109/ICSE.2013.6606550.

[16] S. O. Funtowicz and J. R. Ravetz. *Uncertainty and Quality in Science for Policy*. Google-Books-ID: lINAsNfN7i4C. Springer Science & Business Media, Oct. 31, 1990. 254 pp. ISBN: 978-0-7923-0799-0.

[17] David Garlan. “Software engineering in an uncertain world”. In: *Proceedings of the FSE/SDP workshop on Future of software engineering research*. FoSER ’10. New York, NY, USA: Association for Computing Machinery, Nov. 7, 2010, pp. 125–128. ISBN: 978-1-4503-0427-6. DOI: 10.1145/1882362.1882389. URL: https://dl.acm.org/doi/10.1145/1882362.1882389 (visited on 05/30/2023).

[18] David Garlan. “The Unknown Unknowns Are Not Totally Unknown”. In: *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). ISSN: 2157-2321. May 2021, pp. 264–265. DOI: 10.1109/SEAMS51251.2021.00047.

[19] Gabriel Gehrig. *Data Companion Set - Bachelor’s Thesis - Gabriel Gehrig*. 2023. DOI: 10.5281/zenodo.10034692. (Visited on 10/23/2023).

[20] Sebastian Gerdes, Mohamed Soliman, and Matthias Riebisch. “Decision Buddy: Tool Support for Constraint-Based Design Decisions during System Evolution”. In: *Proceedings of the 1st International Workshop on Future of Software Architecture Design Assistants*. FoSADA ’15. New York, NY, USA: Association for Computing Machinery, May 6, 2015, pp. 13–18. ISBN: 978-1-4503-3438-9. DOI: 10.1145/2751491.2751495. URL: https://dl.acm.org/doi/10.1145/2751491.2751495 (visited on 05/30/2023).

[21] Abhilash Gopalakrishnan and Abhinna Chandra Biswal. “Quiver — An intelligent decision support system for software architecture and design”. In: *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*. 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon). Aug. 2017, pp. 1286–1291. DOI: 10.1109/SmartTechCon.2017.8358574.

[22] Sebastian Hahner. “Dealing with Uncertainty in Architectural Confidentiality Analysis”. In: *Software Engineering (Satellite Events)* (2021).

[23] Sebastian Hahner, Robert Heinrich, and Ralf Reussner. “Architecture-Based Uncertainty Impact Analysis to Ensure Confidentiality”. In: *SEAMS, IEEE/ACM*. 2023, p. 7.

[24] Sebastian Hahner, Robert Heinrich, and Ralf Reussner. *Data Companion Set*. Version 2.0. 2022. DOI: 10.5281/zenodo.6855567. (Visited on 10/18/2023).

[25] Sebastian Hahner et al. “A Classification of Software-Architectural Uncertainty Regarding Confidentiality”. In: *E-Business and Telecommunications*. Ed. by Pierangela Samarati et al. Cham: Springer Nature Switzerland, 2023, pp. 139–160. ISBN: 978-3-031-36840-0.

[26] Sara M. Hezavehi et al. “Uncertainty in Self-adaptive Systems: A Research Community Perspective”. In: *ACM Trans. Auton. Adapt. Syst.* 15.4 (Dec. 20, 2021), 10:1–10:36. ISSN: 1556-4665. DOI: 10.1145/3487921. URL: https://dl.acm.org/doi/10.1145/3487921 (visited on 06/20/2023).

[27] ISO. *Information technology - Security techinques - Information security management systems - Overview and vocabulary (ISO/IEC 27000:2018)*. 2018.

[28] Joel O. Iverson and Robert D. Mcphee. “Knowledge Management in Communities of Practice: Being True to the Communicative Character of Knowledge”. In: *Management Communication Quarterly* 16.2 (Nov. 2002), pp. 259–266. ISSN: 0893-3189, 1552-6798. DOI: 10.1177/089331802237239. URL: http://journals.sagepub.com/doi/10.1177/089331802237239 (visited on 10/21/2023).

[29] A. Jansen and J. Bosch. “Software Architecture as a Set of Architectural Design Decisions”. In: *5th Working IEEE/IFIP Conference on Software Architecture (WICSA’05)*. 5th Working IEEE/IFIP Conference on Software Architecture (WICSA’05). Nov. 2005, pp. 109–120. DOI: 10.1109/WICSA.2005.61.

[30] Stefanie Jasser and Matthias Riebisch. “Reusing security solutions: a repository for architectural decision support”. In: *Proccedings of the 10th European Conference on Software Architecture Workshops*. ECSAW ’16. New York, NY, USA: Association for Computing Machinery, Nov. 28, 2016, pp. 1–7. ISBN: 978-1-4503-4781-5. DOI: 10.1145/2993412.3007556. URL: https://dl.acm.org/doi/10.1145/2993412.3007556 (visited on 05/31/2023).

[31] Afnan Ullah Khan. "Data Confidentiality and Risk Management in Cloud Computing". engd. University of York, Apr. 7, 2014. URL: https://etheses.whiterose.ac.uk/13677/ (visited on 10/05/2023).

[32] P. Layzell, O.P. Brereton, and A. French. "Supporting collaboration in distributed software engineering teams". In: *Proceedings Seventh Asia-Pacific Software Engeering Conference. APSEC 2000*. Seventh Asia-Pacific Software Engineering Conference. ASPEC 2000. Singapore: IEEE Comput. Soc, 2000, pp. 38–45. ISBN: 978-0-7695-0915-0. DOI: 10.1109/APSEC.2000.896681. URL: http://ieeexplore.ieee.org/document/896681/ (visited on 10/21/2023).

[33] Chawanangwa Lupafya. "A conceptual framework for uncertainty in software systems and its application to software architectures". Accepted: 2023-02-06T11:14:14Z. Thesis. The University of St Andrews, June 14, 2023. DOI: 10.17630/sta/264. URL: https://research-repository.st-andrews.ac.uk/handle/10023/26909 (visited on 07/03/2023).

[34] Ioanna Lytra and Uwe Zdun. "Supporting architectural decision making for systems-of-systems design under uncertainty". In: *Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems*. SESoS '13. New York, NY, USA: Association for Computing Machinery, July 2, 2013, pp. 43–46. ISBN: 978-1-4503-2048-1. DOI: 10.1145/2489850.2489859. URL: https://doi.org/10.1145/2489850.2489859 (visited on 07/03/2023).

[35] Steve McConnell. *Software project survival guide*. Redmond, Wash: Microsoft Press, 1998. 288 pp. ISBN: 978-1-57231-621-8.

[36] Klaus North and Gita Kumta. *Knowledge Management*. Springer Texts in Business and Economics. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-59977-9 978-3-319-59978-6. DOI: 10.1007/978-3-319-59978-6. URL: http://link.springer.com/10.1007/978-3-319-59978-6 (visited on 10/26/2023).

[37] Klaus North, Ronald Maier, and Oliver Haas, eds. *Knowledge Management in Digital Change: New Findings and Practical Cases*. Progress in IS. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-73545-0 978-3-319-73546-7. DOI: 10.1007/978-3-319-73546-7. URL: http://link.springer.com/10.1007/978-3-319-73546-7 (visited on 10/21/2023).

[38] Diego Perez-Palacin and Raffaela Mirandola. "Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation". In: *Proceedings of the 5th ACM/SPEC international conference on Performance engineering*. ICPE '14. New York, NY, USA: Association for Computing Machinery, Mar. 22, 2014, pp. 3–14. ISBN: 978-1-4503-2733-6. DOI: 10.1145/2568088.2568095. URL: https://dl.acm.org/doi/10.1145/2568088.2568095 (visited on 05/30/2023).

[39] Oliver Radley-Gardner, Hugh Beale, and Reinhard Zimmermann, eds. *Fundamental Texts On European Private Law*. Hart Publishing, 2016. ISBN: 978-1-78225-864-3 978-1-78225-865-0 978-1-78225-866-7 978-1-78225-867-4. DOI: 10.5040/9781782258674. URL: http://www.bloomsburycollections.com/book/fundamental-texts-on-european-private-law-1 (visited on 10/23/2023).

[40] Andres J. Ramirez, Adam C. Jensen, and Betty H. C. Cheng. "A taxonomy of uncertainty for dynamically adaptive systems". In: *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 2012 7th

International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). ISSN: 2157-2321. June 2012, pp. 99–108. DOI: 10.1109/SEAMS.2012.6224396.

[41] Ralf Reussner, ed. *Modeling and simulating software architectures: the Palladio approach*. Cambridge, Massachusetts: MIT Press, 2016. 377 pp. ISBN: 978-0-262-03476-0.

[42] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empir Software Eng* 14.2 (Apr. 1, 2009), pp. 131–164. ISSN: 1573-7616. DOI: 10.1007/s10664-008-9102-8. URL: https://doi.org/10.1007/s10664-008-9102-8 (visited on 07/03/2023).

[43] Leonie Sterz, Christoph Werner, and Oliver Raabe. "Intelligente Verkehrssysteme – IT-Sicherheit in offenen Infrastrukturen Teil 1". In: *Recht der Datenverarbeitung (RDV) Heft 6 2022* (2022).

[44] Leonie Sterz, Christoph Werner, and Oliver Raabe. "Intelligente Verkehrssysteme – IT-Sicherheit in offenen Infrastrukturen Teil 2". In: *Recht der Datenverarbeitung (RDV) Heft 2 2023* (2023).

[45] Tien Fabrianti Kusumasari et al. "Collaboration model of software development". In: *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*. 2011 International Conference on Electrical Engineering and Informatics (ICEEI). Bandung, Indonesia: IEEE, July 2011, pp. 1–6. ISBN: 978-1-4577-0753-7. DOI: 10.1109/ICEEI.2011.6021769. URL: http://ieeexplore.ieee.org/document/6021769/ (visited on 10/21/2023).

[46] Roger L. Van Scoy. *Software Development Risk: Opportunity, Not Problem:* Fort Belvoir, VA: Defense Technical Information Center, Sept. 1, 1992. DOI: 10.21236/ADA258743. URL: http://www.dtic.mil/docs/citations/ADA258743 (visited on 10/20/2023).

[47] Sitalakshmi Venkatraman and Ramanathan Venkatraman. "Communities of Practice Approach for Knowledge Management Systems". In: *Systems* 6.4 (Sept. 27, 2018), p. 36. ISSN: 2079-8954. DOI: 10.3390/systems6040036. URL: http://www.mdpi.com/2079-8954/6/4/36 (visited on 10/21/2023).

[48] D. Verdon and G. McGraw. "Risk analysis in software design". In: *IEEE Secur. Privacy* 2.4 (July 2004), pp. 79–84. ISSN: 1540-7993, 1558-4046. DOI: 10.1109/MSP.2004.55. URL: https://ieeexplore.ieee.org/document/1324606/ (visited on 10/20/2023).

[49] Gert-Jan de Vreede et al. "Collaboration technology in teams and organizations: Introduction to the special issue". In: *Inf Syst Front* 18.1 (Feb. 1, 2016), pp. 1–6. ISSN: 1572-9419. DOI: 10.1007/s10796-016-9632-3. URL: https://doi.org/10.1007/s10796-016-9632-3 (visited on 10/21/2023).

[50] W.E. Walker et al. "Defining Uncertainty: A Conceptual Basis for Uncertainty Management in Model-Based Decision Support". In: *Integrated Assessment* 4.1 (Mar. 1, 2003). Publisher: Taylor & Francis _eprint: https://doi.org/10.1076/iaij.4.1.5.16466, pp. 5–17. ISSN: 1389-5176. DOI: 10.1076/iaij.4.1.5.16466. URL: https://doi.org/10.1076/iaij.4.1.5.16466 (visited on 05/30/2023).

[51] Etienne Wenger, Richard A. McDermott, and William Snyder. *Cultivating communities of practice: a guide to managing knowledge*. Boston, Mass: Harvard Business School Press, 2002. 284 pp. ISBN: 978-1-57851-330-7.

[52] Danny Weyns et al. "Towards a Research Agenda for Understanding and ManagingUncertainty in Self-Adaptive Systems". In: *SIGSOFT Softw. Eng. Notes* 48.4

(Oct. 13, 2023), pp. 20–36. ISSN: 0163-5948. DOI: 10.1145/3617946.3617951. URL: https://dl.acm.org/doi/10.1145/3617946.3617951 (visited on 11/03/2023).

[53] Tao Yue. *Precise Semantics for Uncertainty Modeling (PSUM)*. Version 1.0. 2023. URL: https://www.omg.org/spec/PSUM/1.0/.

[54] Olaf Zimmermann et al. "Reusable Architectural Decision Models for Enterprise Application Development". In: *Software Architectures, Components, and Applications*. Ed. by Sven Overhage et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 15–32. ISBN: 978-3-540-77619-2. DOI: 10.1007/978-3-540-77619-2_2.