S. Hahner, N. Niehues, N. Boltz, M. Fuksa, and R. Heinrich, "ARC³N: A Collaborative Uncertainty Catalog to Address the Awareness Problem of Model-Based Confidentiality Analysis," in ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24), ACM, 2024, doi: 10.1145/3652620.3688556

# ARC³N: A Collaborative Uncertainty Catalog to Address the Awareness Problem of Model-Based Confidentiality Analysis

Sebastian Hahner
Karlsruhe Institute of Technology
Karlsruhe, Germany
sebastian.hahner@kit.edu

Nils Niehues
Karlsruhe Institute of Technology
Karlsruhe, Germany
nils.niehues@kit.edu

Nicolas Boltz
Karlsruhe Institute of Technology
Karlsruhe, Germany
nicolas.boltz@kit.edu

Mario Fuksa
University of Stuttgart
Stuttgart, Germany
mario.fuksa@iste.uni-stuttgart.de

Robert Heinrich
Karlsruhe Institute of Technology
Karlsruhe, Germany
robert.heinrich@kit.edu

## Abstract

Identifying confidentiality violations is challenging as modern software-intensive systems exchange and store large amounts of data, and system deployment and context vary. Although model-based analyses can identify such violations already at design time, uncertainty within a software system or its environment can void analysis results. Existing approaches to raising awareness of uncertainty sources are limited in usability and extendability and require expert knowledge for interpretation and analysis. This paper presents our collaborative tooling ARC³N for collecting, modeling, and analyzing uncertainty sources regarding confidentiality. Using an open web-based platform, we simplify both identifying and assessing uncertainty without requiring expert knowledge. We evaluate our approach with a user study with students, researchers, and practitioners (n = 17) and demonstrate its feasibility.

## CCS Concepts

• **Software and its engineering → Model-driven software engineering**; **Software design engineering**.

## Keywords

Model-driven Security, Software Architecture, Confidentiality, Uncertainty, Unknown Unknowns, Uncertainty Awareness Problem

**ACM Reference Format:**
Sebastian Hahner, Nils Niehues, Nicolas Boltz, Mario Fuksa, and Robert Heinrich. 2024. ARC³N: A Collaborative Uncertainty Catalog to Address the Awareness Problem of Model-Based Confidentiality Analysis. In *ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24), September 22–27, 2024, Linz, Austria.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3652620.3688556

## 1 Introduction

Software engineering is complex, and uncertainty is uncertain. With the growing interconnection of modern systems, building secure software becomes increasingly difficult. This challenge is particularly acute regarding confidentiality [23] that ensures that information is not disclosed to unauthorized entities [15]. Accidental data breaches and cyber-attacks are on the rise [5]. Uncertainty, the lack of deterministic knowledge about a system [19], complicates the analysis and mitigation of confidentiality violations.

Existing model-based confidentiality analyses under uncertainty [12] require known uncertainty sources, also called first-order uncertainty [19], such as variations in sensors or user behavior. However, second-order uncertainty, or *unknown unknowns*, cannot be analyzed due to the lack of knowledge about its sources, limiting the development of mitigation strategies [29]. This *uncertainty awareness problem* impedes comprehensive software system analysis, posing security threats that could be resolved at design time. Put simply, one cannot analyze what one does not know.

The *uncertainty awareness problem* is supported in the literature, highlighting the challenge of unanticipated change [28], the need to share uncertainty knowledge [14], and uncertainty management [29]. Approaches include classifications [11, 19], surveys [20, 26], and reusing expert knowledge [18]. However, classifications only describe properties, and relying on expert knowledge is impractical. Addressing this problem enables end-to-end approaches with a higher impact than incremental analysis enhancements [29].

This paper presents the tool ARC³N to tackle the *uncertainty awareness problem* by collecting uncertainty sources in software architectures. Based on a previously developed classification [11], we create a web-based tool to support the community in organizing identified uncertainty sources, addressing the drawbacks of publication-based collections or checklists. It simplifies identifying and assessing uncertainty sources in model-based confidentiality analyses, enables cooperation between researchers and practitioners, and overcomes the limitation of scattered knowledge among researchers and institutions. We foster two types of collaboration: ARC³N benefits the communication between different roles with different knowledge, e.g., security experts and software architects. It also benefits peers with the same role as a shared catalog for exchange and documentation. We evaluate ARC³N in two ways: First, we conduct a study with 17 participants (students, researchers, and practitioners) of varying expertise. Participants use our tool to perform tasks such as describing and identifying uncertainty sources in software architectures to assess if the tool supports this with little or no prior knowledge. Second, we extend an existing analysis [10], which previously suffered from the *uncertainty awareness problem*.


*MODELS Companion '24, September 22–27, 2024, Linz, Austria*

ACM ISBN 979-8-4007-0622-6/24/09
https://doi.org/10.1145/3652620.3688556

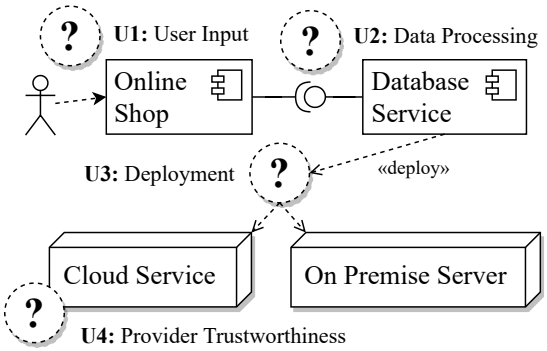


**Figure 1: Running example based on a simplified online shop with four uncertainty sources highlighted**

## 2 Foundations and Running Example

To better understand the dimensions and properties of uncertainty, various classifications have been proposed, e.g., in the domain of self-adaptive systems [14] or cyber-physical systems [1]. Common dimensions of these classifications are the location, level, nature, manageability, and impact on quality attributes. In this paper, we reuse a classification that is tailored to confidentiality [11]. This classification focuses on representing uncertainty in software architectural models and consists of 8 categories with 27 options. By classifying uncertainty according to these categories, software architects become aware of uncertainty's impact on confidentiality, which helps in analysis and mitigation.

We introduce the running example of a simple online shop. Similar examples have been used in previous work [4, 11, 12]. Figure 1 shows the online shop's combined component and deployment diagram. A user accesses the *Online Shop* component to buy a product. This action involves personal data and thus comes with confidentiality requirements, e.g., related to the encryption and storage in the *Database Service* component. Here, multiple uncertainty sources arise: The user input (**U1**) is a type of *human in the loop* [19], both the data processing between the components (**U2**) and the deployment location of the database (**U3**) are open architectural design decisions, and the provider's trustworthiness (**U4**) is part of the system's context. According to the classification in [11], the provider trustworthiness (**U4**) can be classified as environmental, external scenario uncertainty. It is partially reducible because appropriate measures exist to minimize its impact, but it only fully resolves at runtime. It only has an indirect but high impact on confidentiality, as it criticality depends on other design issues or uncertainties, e.g., a lack of encryption as part of (**U2**).

Depending on their outcome, all uncertainties can void previous confidentiality analysis results, e.g., because unencrypted data flows to the cloud of a non-trusted provider. To mitigate these uncertainties, software architects must first be aware of their sources.

## 3 A Web-Based Collaborative Approach

In this section, we present our approach to address the *uncertainty awareness problem*. We discuss identified problems and resulting requirements. Then, we present the metamodel and its realization.

Related approaches like uncertainty classification [11, 20], the collection of architectural knowledge [8], or privacy patterns [6] have several shortcomings regarding usability, extensibility, and longevity of the contained information. We outline four key requirements for ARC$^3$N to address these shortcomings. **R1**: Classifications should include relations and inheritance between uncertainty sources, with the ability to filter, search, and navigate, due to the dependencies between uncertainties [7, 11]. **R2**: Explainability should be emphasized, utilizing examples, interactive visualizations, and context information to provide comprehensive understanding [3]. **R3**: An open-source and publicly available catalog simplifies the discussion and extension by researchers and practitioners, addressing the issues of outdated, incomplete, or unavailable research artifacts [6, 8, 16]. **R4**: To facilitate automated analyses and end-to-end approaches, catalogs should be transformable into machine-readable formats like JSON or XML.

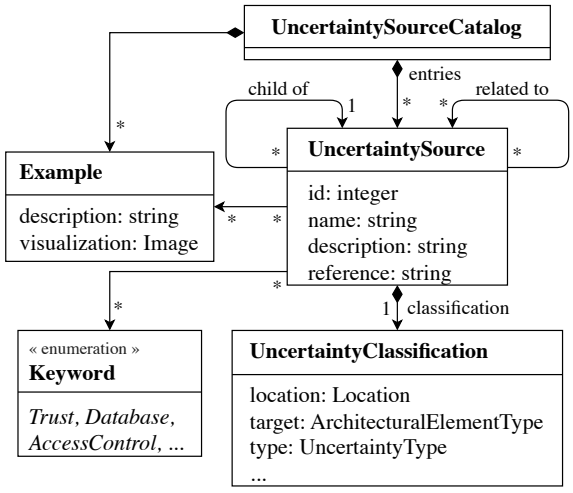


**Figure 2: Metamodel of the uncertainty source catalog**

### 3.1 Metamodel for Uncertainty Sources

As discussed above, classifications alone cannot provide enough context information for software architects. Figure 2 shows the simplified metamodel of an *Uncertainty Source Catalog* that addresses requirements **R1** and **R2**. The *Uncertainty Source Catalog* consists of *Uncertainty Sources* and *Examples* that demonstrate these sources. Each source consists of a unique id, a name, a description, and a literature reference for additional information about its origin. It can be related to an arbitrary number of other sources and can have children representing inheritance between sources. To classify the source, it is described by an *UncertaintyClassification* using the classification [11] presented in Section 2. Last, sources reference appropriate *Examples* and can be further described by *Keywords* to group uncertainty sources beyond their classification.

In our running example presented in Section 2, uncertainty **U4** describes the unknown trustworthiness of the cloud service provider. This uncertainty source is part of the initial population of our catalog based on related work [11, 20] and tagged with *Trust*. Besides its own properties like an expressive description and its classification (see Section 2), the catalog exemplifies this uncertainty with confidentiality questions regarding governmental institutions influencing cloud service providers. The uncertainty of *Provider Trustworthiness* is a child of the uncertainty supertype *Unpredictable*

*Environment* [20] and related to uncertainty sources of our initial population regarding encrypted communication, component trustworthiness, and deployment location.

This information provides comprehensive context knowledge for software architects, raises awareness, and simplifies the assessment. By attaching this information and also linking related uncertainty types, we address the organization issue (**R1**) and enable to search and filter uncertainty sources. Additionally, there are no space limitations regarding examples, visualizations, and discussions (**R2**). To demonstrate this, we enriched 51 uncertainty sources from multiple classifications [11, 20] with context information, examples, and also uncertainty relations and inheritance as part of our data set [13].

## 3.2 Realization as Web-Based Catalog

Collecting uncertainty sources only within publications [20] or data sets [11] lacks reusability, extensibility, and availability, as discussed above. To address requirements **R3** and **R4**, we realized our metamodel and catalog approach as an open-source and ready-to-use collaborative tool that is publicly available[1], named *Rese**arch** **arc**hive for Software-**Arc**hitectural U**n**certainty*, short **ARC³N**.

Our tooling revolves around a catalog of uncertainty sources and their classification, allowing for quick navigation, search, and filtering by keywords or classification categories. Software architects can select a source to view its description, examples, related sources, and classification. Because there is no space limitation, the selected options of the classification can be explained in detail. We also provide further explanation of all classification categories and relate known uncertainty sources. Last, the full catalog can be downloaded in a machine-readable JSON format.

We chose a web-based approach to simplify collaboration. Our backend, a GitHub repository, allows adding and discussing new uncertainty sources with participants. These sources are added via our tool and retrieved using the open GitHub API, ensuring lightweight and accessible information storage. GitHub reduces the risk of link decay common on institutional websites [9]. Information retrieval is automated using a GitHub Actions-based DevOps pipeline. Once updated, our tool operates autonomously, supporting self-hosting or local use, and facilitates easy integration into other platforms.

ARC³N collects uncertainty sources in a central place instead of relying only on publication-based collections, simplifying the collaboration of researchers and practitioners (**R3**). Everyone can propose new uncertainty sources based on the open-source GitHub repository. All required information is stored as part of the tool support and can be downloaded in third-party analyses (**R4**). In sum, our approach tackles the aforementioned requirements (**R1** – **R4**) and provides an easy-to-use, extensible, and open collaborative approach to address the lack of awareness regarding uncertainty sources in architectural confidentiality analysis.

# 4 User Study-based Evaluation

To evaluate our approach, we conducted a user study involving 17 participants. In this section, we show the evaluation plan and design, and we present and discuss the evaluation results. We use a Goal-Question-Metrics-plan [2] to face the lack of guideline-based evaluation in current software architecture research [16]. As our approach aims to assist software architects in identifying and understanding uncertainty sources, our goal is to evaluate our approach's suitability and usability for this endeavor (**G1**). We subdivide this goal into three questions: **Q1**: Can our approach help identify and describe uncertainty sources in software architectures? **Q2**: Does the approach support collaboration and discussion? **Q3**: Is our tool easy to use and provides a good user experience?

Answering the first question **Q1** is crucial for evaluating whether our approach helps in addressing the *uncertainty awareness problem* and our requirements **R1** and **R2**, see Section 3. We measure the correctness of the participants' answers (**M1**) by comparing them to a gold standard and rating them as correct or incorrect. The second question **Q2** targets the collaboration aspect and requirement **R3**. We require the participants to argue based on a discussion found online and rate their reasoning (**M2**) similar to the first metric. The third question focuses on our realization of ARC³N. Here, we apply the standardized System Usability Scale (SUS) [17] for the measurement (**M3**) and ask about the usefulness of single features (**M4**) to get more precise results. Please note, that evaluating requirement **R4** (machine-readability) is not part of the user study but of the demonstration presented thereafter.

## 4.1 User Study Design

Our user study consists of a self-assessment, an introduction to uncertainty in software architecture and our tooling, two assignments, a SUS, and a feedback session. Participants first self-assess their expertise in areas like software architecture and uncertainty, aiding in result interpretation. They then receive a three-page introduction and a walkthrough of our tool ARC³N.

To address **Q1** and **Q2**, participants complete two assignments. Each assignment includes an architecture diagram and scenario description. In the first assignment (**A1.1**), they map textual descriptions of uncertainties to sources in our catalog. In the second part (**A1.2**), they evaluate the appropriateness of an uncertainty description. The first task assesses search and navigation (**M1**), while the second includes context information and discussion aspects (**M2**). The second assignment (**A2**) requires identifying uncertainties in the architecture diagram without hints using our tool support (**M1**).

For **Q3**, participants complete a SUS (**M3**) with ten questions on a five-point scale. In the feedback session, they rate the usefulness and intuitiveness (**M4**) of ARC³N's features and discuss their learning in the self-assessed knowledge areas, providing additional qualitative feedback. To avoid fatigue, sessions lasted no longer than one hour.

## 4.2 Evaluation Results and Discussion

We conducted the user study with a total of 17 participants, including 7 students, 5 researchers, and 5 practitioners. All participants had at least basic knowledge of software architecture and design decisions. Regarding uncertainty, confidentiality, and security analysis, the expertise is more evenly spread, with only very few participants considering themselves experts. We conclude that these are excellent conditions, as our approach shall not require expert knowledge but help beginners and intermediate users. After we received all 17 participants' results, we evaluated them by comparing them to our gold standard, which was created independently

[1] Available in our data set [13] and also online: https://arc3n.abunai.dev/

**Table 1: Percentage of correct answers by prior knowledge**

| Uncertainty Knowledge | A1.1 | A1.2 | A2 |
|---|---|---|---|
| No Prior Knowledge | 0.500 | 0.600 | 0.923 |
| Little Prior Knowledge | 0.929 | 0.429 | 0.944 |
| Good Prior Knowledge | 0.875 | 1.000 | 1.000 |
| Expert Prior Knowledge | 1.000 | 1.000 | 1.000 |

and then unified by two researchers. Also, two researchers conducted the comparison independently, and all inconsistencies were discussed and resolved. In the following, we present the results for the evaluation questions **Q1** – **Q3**.

The first question (**Q1**) addresses the identification of uncertainty, measured by the correctness of participants' answers (**M1**) for tasks **A1.1** and **A2**. Table 1 shows the percentage of correctly identified uncertainty sources, grouped by prior knowledge. Correct answers ranged from 50% to 100% in task **A1.1** and from 92% to 100% in task **A2**. Notably, correctness in task **A1.1** increased from 50% to nearly 93%, indicating our tool helps identify uncertainties even with minimal knowledge. In task **A2**, all participants, regardless of prior knowledge, achieved at least 92% correctness. This improvement may be due to learning effects, as participants became more proficient with the tool. The overall average correctness across tasks is 87.35%, compared to 73% in a similar study without tool support [11]. Thus, our approach aids in identifying relevant uncertainty sources, particularly for non-experts.

The second question **Q2** asks about collaboration and discussion. To answer this question, the participants had to read context information and discussions about uncertainty and provide their own reasoning in task **A1.2**, for which the correctness was then evaluated (**M2**). Table 1 shows a larger gap between participants with no or little knowledge and more experienced users. One possible explanation could be that the earlier prototype used for this assignment did not highlight discussions enough, see [13]. Nevertheless, the average correctness reaches 65%.

The third question (**Q3**) concerns the usability of our tool, evaluated using a SUS (**M3**) and ratings for the usefulness and intuitiveness of features on a scale from 1 to 4 (**M4**). The cumulative SUS score of 69.7 indicates good usability, comparable to a previous study's score of 68.3 [11]. Participants appreciated explanations with example scenarios and graphics, scoring an average of 3.76 out of 4. The overall usefulness averaged 3.2, and intuitiveness averaged 3.0. Based on individual results and qualitative feedback, we further improved the visualization quality of ARC$^3$N, see [13].

At the end of the user study, we asked the participants about their learnings in different categories on a scale from 1 to 4. The participants stated that they had learned the most about uncertainty and the different uncertainty types, with scores of 3.4 and 3.7, respectively. All qualitative feedback can be found in the data set [13]. We conclude that this represents a promising result as it indicates that our approach increased both knowledge and awareness, thus addressing the *uncertainty awareness problem*.

## 4.3 Threats to Validity

Following Runeson and Höst [22], we identify the main threat to internal validity as the potential variability in individual researchers' assessments of participants' answers, which we address through structured sessions and independent reviews by two researchers.

**Listing I: Input and results of the extended impact analysis**

```
1  Enter an id of an uncertainty to check for:
2  > #48
3
4  Select one of these elements.
5  1) "On Premise Server" (_Qh8vMLB7E6PGKdWgLSwQQ)
6  2) "EU Cloud Service" (_8BEvUB-Ee6TRb5PGryaNg)
7  3) "Non-EU Cloud Service" (_1JoBkITy2mO_IpTxeAg)
8
9  Enter line number:
10 > 3
11
12 Analysis completed. Result:
13 External uncertainty impact found: StartAction
14 (_yVL18ITkEe2mO_IpTxeAg) of service "storeData"
15
16 Impact Set (size 11): ...
```

Threats to external validity arise from the use of convenience sampling, the small sample size, and the lack of a control group, which may limit the generalizability of the findings. However, these issues were mitigated by including participants from diverse occupational backgrounds and varying levels of knowledge, thereby enhancing the study's representativeness [11, 20]. Construct validity was addressed through a GQM-based evaluation plan and structuring the user study similar to a previous study [11], while the publication of ARC$^3$N and all raw evaluation data [13] simplifies the repeatability of our study to enhance the reliability of our findings.

## 5 Demonstration for Impact Analysis

In the second part of our evaluation, we demonstrate the feasibility of applying ARC$^3$N. We apply it to an existing uncertainty impact analysis [10] that suffers from the *uncertainty awareness problem* like many other uncertainty-aware analyses [14]. In the following, we summarize the analysis and the integration of our catalog.

The architecture-based impact analysis helps identify potential confidentiality violations early at design time [10] to understand the effects of uncertainty better. Similar to change impact analysis, this analysis returns an impact set of potentially affected elements of the modeled software architecture. This is achieved by propagating uncertainty sources first within the collaboratively created architectural models and then in extracted data flow diagrams.

However, such analysis requires software architects to identify uncertainty sources, classify them by themselves, and annotate them to the correct element of software architecture. This not only requires expert knowledge [11, 12] but also awareness about relevant uncertainty sources and their classification. Failing to correctly identify, classify, and annotate severely limits analysis results' validity, as they can lack both precision and comprehensiveness.

We addressed this limitation by connecting the impact analysis to our catalog, thereby demonstrating its feasibility. Listing I shows how the interaction can look like using an interactive command prompt. The analysis loads the architectural models [21] and the serialized catalog of uncertainty sources from ARC$^3$N. Afterward, software architects are asked to provide the id of the uncertainty they want to analyze which can be retrieved from the catalog. The extended analysis queries the uncertainty source's classification and iterates over the loaded models to identify matching elements. Examples are behavior descriptions using *SetVariableActions* for

behavioral uncertainty, or hardware resources represented by *ResourceContainers* for external uncertainty like the provider's trustworthiness (**U4**) from the running example. Last, software architects can choose the elements to be annotated. The analysis starts the automated propagation of uncertainty and yields the results.

The extension was realized as an additional step prior to the impact analysis process. This addresses the *uncertainty awareness problem* by providing software architects with a comprehensive list of possible uncertainty sources to choose from. Additionally, less expert knowledge is required about the classification categories and their options as the interpretation is done automatically. Last, the manual effort of identifying potential elements to annotate is reduced. In sum, this demonstrates the feasibility of extending an existing analysis using our approach without altering the core propagation algorithm. We stress that other uncertainty-aware analysis [12, 27] could similarly benefit from such an extension.

## 6 Related Work

This section summarizes related work relevant to this paper:

*Uncertainty Management in Software Systems:* Recent work underscores the need for uncertainty management [28] and reusable methods [29], with many classifications [11, 19, 20] and surveys [14, 26] providing a foundation. However, comprehensive tool support and confidentiality focus are lacking. ARC³N addresses these gaps and can be easily integrated into existing analysis methods.

*Collaborative Approaches in Knowledge Management:* Related work shows the need for collaborative knowledge sharing to address uncertainty [14], with examples from legal sciences [25] and tooling regarding design decisions [8] and privacy patterns [6]. However, these approaches are unsuitable for addressing uncertainty and confidentiality due to domain gaps and issues like link decay [9]. ARC³N addresses these issues, ensuring usability and longevity.

*Software Architecture and Design Decisions:* Especially at design time, uncertainty is related to architectural design decisions [11]. Comprehensive approaches for modeling and analysis of software architecture models exist [21], also with relation to uncertainty [24]. These approaches do not focus on confidentiality, and although uncertainty-aware and model-based confidentiality analyses exist [12, 27], they still suffer from the *uncertainty awareness problem*. Our tool integrates uncertainty management into modeling and analysis tools, enabling software architects to address confidentiality in uncertain environments, and also enhancing explainability [3].

## 7 Conclusion

This paper presented a collaborative, web-based tool for collecting and organizing uncertainty sources based on existing classifications [11, 20], extending beyond publication-based collections. ARC³N contains 51 uncertainty sources, is open-source, publicly available, and is easy to incorporate into comprehensive frameworks. We demonstrated this by extending an existing model-based analysis [10], effectively addressing the *uncertainty awareness problem*. We also evaluated our tool with a user study comprising 17 participants from industry and academia. The results indicate a high correctness of 87% in identifying uncertainty in software architecture while also increasing both awareness and knowledge about uncertainty. In future work, we build on ARC³N to explore further approaches regarding the *uncertainty awareness problem*, e.g., interactive guides, recommender systems, automatic model repair, or applying large language models to identify and explain uncertainty.

## Acknowledgments

This is partially based on the research project SofDCar (19S21002), which is funded by the German Federal Ministry for Economic Affairs and Climate Action, and supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs, the BMBF (German Federal Ministry of Education and Research) grant number 16KISA086 (ANYMOS), and the NextGenerationEU project by the European Union (EU). We like to thank Gabriel Gehrig and Alexander Vogt.

## References

[1] Maribel Acosta et al. 2022. Uncertainty in coupled models of cyber-physical systems. In *MODELS-C*.
[2] Victor R. Basili et al. 1984. A Methodology for Collecting Valid Software Engineering Data. *IEEE TSE* (1984).
[3] Marcello M. Bersani et al. 2023. A Conceptual Framework for Explainability Requirements in Software-Intensive Systems. In *REW*.
[4] Nicolas Boltz et al. 2024. An Extensible Framework for Architecture-Based Data Flow Analysis for Information Security. In *ECSA*.
[5] Bundeskriminalamt. 2021. Cybercrime Bundeslagebild.
[6] Michael Colesky and Julio C. Caiza. 2018. A System of Privacy Patterns for Informing Users: Creating a Pattern System. In *EuroPLoP*.
[7] Javier Cámara et al. 2024. Uncertainty Flow Diagrams: Towards a Systematic Representation of Uncertainty Propagation and Interaction in Adaptive Systems. In *SEAMS*.
[8] Sebastian Gerdes et al. 2015. Decision Buddy: Tool Support for Constraint-Based Design Decisions during System Evolution. In *FoSADA*.
[9] Dion Hoe-Lian Goh and Peng Kin Ng. 2007. Link decay in leading information science journals. *JASIST* (2007).
[10] Sebastian Hahner et al. 2023. Architecture-Based Uncertainty Impact Analysis to Ensure Confidentiality. In *SEAMS*.
[11] Sebastian Hahner et al. 2023. A Classification of Software-Architectural Uncertainty Regarding Confidentiality. In *ICETE*.
[12] Sebastian Hahner et al. 2023. Model-based Confidentiality Analysis under Uncertainty. In *ICSA-C*.
[13] Sebastian Hahner et al. 2024. *ARC³N - Data Set.* https://doi.org/10.5281/zenodo.11186636
[14] Sara M. Hezavehi et al. 2021. Uncertainty in Self-adaptive Systems: A Research Community Perspective. *TAAS* (2021).
[15] International Organization for Standardization. 2018. ISO 27000:2018.
[16] Marco Konersmann et al. 2022. Evaluation Methods and Replicability of Software Architecture Research Objects. In *ICSA*.
[17] James R Lewis. 2018. The system usability scale: past, present, and future. *International Journal of Human–Computer Interaction* (2018).
[18] Ioanna Lytra and Uwe Zdun. 2013. Supporting architectural decision making for systems-of-systems design under uncertainty. In *SESoS*.
[19] Diego Perez-Palacin and Raffaela Mirandola. 2014. Uncertainties in the modeling of self-adaptive systems. In *ICPE*.
[20] Andres J. Ramirez et al. 2012. A taxonomy of uncertainty for dynamically adaptive systems. In *SEAMS*.
[21] Ralf H Reussner et al. 2016. *Modeling and simulating software architectures: The Palladio approach.* MIT Press.
[22] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *ESE* (2009).
[23] Stephan Seifermann et al. 2022. Detecting violations of access control and information flow policies in data flow diagrams. *JSS* (2022).
[24] Dalia Sobhy et al. 2021. Evaluation of Software Architectures under Uncertainty: A Systematic Literature Review. *TOSEM* (2021).
[25] Leonie Sterz et al. 2022-2023. Intelligente Verkehrssysteme – IT-Sicherheit in offenen Infrastrukturen. *Recht der Datenverarbeitung* (2022-2023).
[26] Javier Troya et al. 2021. Uncertainty representation in software models: a survey. *Software and Systems Modeling* (2021).
[27] Maximilian Walter et al. 2022. Architectural Optimization for Confidentiality Under Structural Uncertainty. In *ECSA*.
[28] Danny Weyns. 2020. *An introduction to self-adaptive systems: A contemporary software engineering perspective.* John Wiley & Sons.
[29] Danny Weyns et al. 2023. Towards a Research Agenda for Understanding and Managing Uncertainty in Self-Adaptive Systems. *SEN* (2023).