M. Walter et al., "Architectural Optimization for Confidentiality Under Structural Uncertainty," in European Conference on Software Architecture (ECSA), Springer, 2022, doi: 10.1007/978-3-031-15116-3_14

Architectural Optimization for Confidentiality under Structural Uncertainty*

Maximilian Walter[1], Sebastian Hahner[1], Stephan Seifermann[1], Tomas Bures[2], Petr Hnetynka[2], Jan Pacovský[2], and Robert Heinrich[1]

[1] KASTEL – Institute of Information Security and Dependability, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{maximilian.walter,sebastian.hahner,stephan.seifermann,robert.heinrich}@kit.edu
[2] Charles University, Czech Republic
{bures,hnetynka,pacovsky}@d3s.mff.cuni.cz

Abstract. More and more connected systems gather and exchange data. This allows building smarter, more efficient and overall better systems. However, the exchange of data also leads to questions regarding the confidentiality of these systems. Design notions such as Security by Design or Privacy by Design help to build secure and confidential systems by considering confidentiality already at the design-time. During the design-time, different analyses can support the architect. However, essential properties that impact confidentiality, such as the deployment, might be unknown during the design-time, leading to structural uncertainty about the architecture and its confidentiality. Structural uncertainty in the software architecture represents unknown properties about the structure of the software architecture. This can be, for instance, the deployment or the actual implementation of a component. For handling this uncertainty, we combine a design space exploration and optimization approach with a dataflow-based confidentiality analysis. This helps to estimate the confidentiality of an architecture under structural uncertainty. We evaluated our approach on four application examples. The results indicate a high accuracy regarding the found confidentiality violations.

Keywords: Uncertainty · Confidentiality · Design Space Exploration · Software Architecture · Access Control · Information Flow

1 Introduction

The gathering of more data and, at the same time, the exchange of existing data enables more efficient and smarter systems. This enables, for instance, the

* This work was supported by the German Research Foundation (DFG) under project number 432576552, HE8596/1-1 (FluidTrust), as well as by funding from the topic Engineering Secure Systems (46.23.03) of the Helmholtz Association (HGF) and by KASTEL Security Research Labs. Additionally, it was supported by the Czech Science Foundation project 20-24814J, and also partially supported by Charles University institutional funding SVV 260451.

digitization of the production process [21] or eHealth services. These systems often include different stakeholders such as customers, suppliers, or public service providers. Each of these stakeholders has different systems and often different regulations. However, they exchange data and, therefore, build a complex network. Nevertheless, this exchange shall still keep the confidentiality of the involved stakeholders' data. Confidentiality is a part of information security. It is the "property that information is not made available or disclosed to unauthorized individuals, entities, or processes" [24]. It is also part of privacy such as in the General Data Protection Regulation (GDPR) [10]. However, privacy often contains more aspects like the right to forget or delete data. Violations against the GDPR can have high costs [40]. Understanding these complex networks and finding confidentiality violations is very difficult. In addition, confidentiality is not the only relevant quality of a system. Other quality attributes such as costs or performance also need to be considered, and this consideration of multiple different attributes complicates the development further.

Approaches such as Security by Design or Privacy by Design want to tackle these confidentiality issues. They propose to continuously consider the security concern in all phases of the software development [45]. These phases include especially the design-time. Additionally, finding and repairing flaws in earlier phases can reduce later costs significantly [4]. This can also be seen in cohesion with the new "Insecure Design" [37] category from the OWASP10 [38] list. This list contains the top 10 categories of security problems for web applications. Therefore, analyzing the system during design-time for confidentiality issues might reduce this type of confidentiality problem. However, not all decisions are already made during the design-time, and some remain unclear. Especially, the impact on confidentiality is unknown [19]. For instance, the concrete deployment might be unknown during the early design-time, e.g., cloud vs on-premise or within EU or outside EU. Therefore, an uncertainty exists regarding the future confidentiality of the system. In a previous publication, we classified uncertainty [6] regarding the confidentiality of software architectures. Based on this classification, we would classify the deployment as structural uncertainty. This paper is the first step to handle uncertainty and confidentiality during the design-time. Therefore, we choose first to consider structural uncertainty since it can be represented by the creation of different architecture variations. For this variation creation, there exists already approaches such as design space exploration approaches. In another previous publication, we already applied design-time confidentiality analysis for handling environmental uncertainty [5]. However, this approach is not immediately applicable to handle structural properties of software architectures.

Our contribution for tackling this problem is the combination of a design space exploration, and optimization approach [30] together with a dataflow-based confidentiality analysis [49]. This goes beyond the pure confidentiality analysis [47] by explicitly considering uncertainty. Using this combination, architects can first model different design decisions. These design decisions are then used to automatically create different architecture variations, which are analyzed for confidentiality. Since we reuse an existing approach for the design

space exploration [30], our approach can consider, besides the newly added confidentiality, additional quality attributes, such as costs, and calculate the Pareto optimal candidate regarding all considered quality metrics. This optimization enables software architects to make informed trade-off decisions between different quality metrics.

We evaluated our approach based on four application examples for feasibility and accuracy regarding the consideration of confidentiality violations. All application examples are based on either the design space exploration or the confidentiality research domain. The results for these examples indicate that our approach is feasible and has a high accuracy regarding the detection of confidentiality violations in our examples.

The paper is structured as follows. We first introduce our running example in Section 2. Afterwards, we describe in Section 3 our foundations. Then, we describe the used confidentiality analysis in Section 4. In Section 5, we describe our new design space exploration integration. The evaluation follows in Section 6. In Section 7, we describe related work and Section 8 concludes the paper.

2 Running Example

We illustrate our approach by using a running example based on [32,18]. It represents a simplified online shop inside the European Union (EU) with EU customers. The example consists of two components and two deployment locations, shown in Figure 1. The `Online Shop` component provides the basic shop system and is deployed on an `On Premise Server` within the EU. The `Database Service` component that persists data can either be deployed on the same EU-based server or a `Cloud Service` outside the EU. Both deployments are technically feasible.

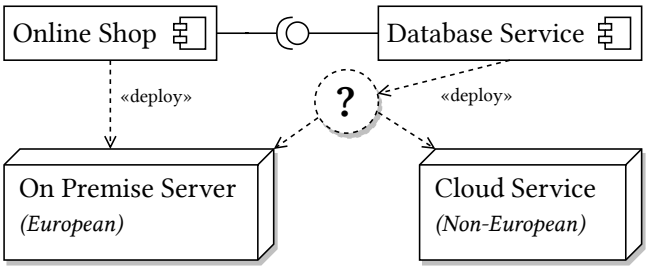

Fig. 1. Component and deployment diagram under uncertainty (denoted by `"?"`)

However, they might not be legally possible since we need to consider the sensitivity of the data. The GDPR [10] forbids to transfer personal data outside of the EU except several conditions are met. Thus, if the `Database Service` is used to store personal data, the component cannot be deployed outside the EU.

Therefore, a design decision regarding the allocation is necessary. This decision is affected by the type of the transferred data. Leaving this architectural design decision open introduces uncertainty about the structure of the system. In the worst case, this introduces confidentiality issues that violate the GDPR. This design decision is a common design decision companies face today if they consider moving from on premise solutions to cloud-based solutions. Note that this example is very basic, and the design decisions are mostly obvious. However, it helps to describe and easily understand the involved activities. The approach itself can be also used for more complicated systems that are subject to a multitude of different design decisions like multiple deployment questions or to compare different implementations.

3 Foundations

For modelling the software architecture, we use the Architecture Description Language (ADL) Palladio [43]. For the optimization of the architecture regarding different quality metrics, we use PerOpteryx [30].

3.1 Palladio

We choose the Palladio Component Model (PCM) [43] as our ADL since there exists already an optimization process, and it supports various quality analyses such as performance, reliability, and costs. Additionally, it has a well-established tooling support, which is freely available. PCM is developed for the component-based development process and consists of five different models. Architects can model components and their required and provided interfaces in the repository model. The interfaces specify services with parameters and return values. These services are implemented by components with the so-called ServiceEffectSpecifications (SEFF). Different components can be instantiated and connected in the system (assembly) model. Additionally, the public interfaces of the system are instantiated there. The services of these public interfaces are the services that directly interact with the user. The user behavior is described in the usage model. Also, PCM models the deployment of components. Here, the resource environment describes hardware resource containers such as servers and other computing devices and linking hardware resources such as network nodes (e.g., switches). The allocation model specifies the placement of components to particular hardware resources.

3.2 PerOpteryx

PerOpteryx [30] is a design space exploration and optimization approach for PCM. It supports the generation of different architectural variations and the optimization for different quality attributes. It uses evolutionary search algorithms and calculates the Pareto optimal architectural variation. The different variation points of the architecture are specified in the design decision model.

Here, architects can define which entities in the software architecture can be configured. For instance, in our running example, this is the allocation of the `Database Service` component to the EU and the non-EU servers. The different quality attributes are defined as quality dimensions. There exist various quality dimensions, such as costs or performance.

However, there is no quality dimension for confidentiality, which would be required for our running example. The optimization approach is guided by the Quality of Service Modeling Language (QML) [15] contracts [36]. Here, architects can specify which quality dimension should be optimized and what the desired values should be. Additionally, they can specify restrictions for the optimization, e.g., that the costs cannot be higher than a certain threshold. Each generated and evaluated variation is called a candidate. Besides the optimal candidates, PerOpteryx also yields all the investigated candidates.

4 Architecture-Centric Confidentiality Analysis

For determining the confidentiality of a system, an automatic analysis based on the properties of the system, such as deployment locations or user roles, is beneficial. With this analysis, architects can analyze different scenarios regarding their confidentiality. For instance, based on our running example, an architect can model the scenario using the EU server first and the scenario using a server outside of the EU afterwards. Both scenarios could be analyzed regarding confidentiality, and afterwards, the most beneficial scenario could be chosen. To support this use case, we need to model the different scenarios and analyze the modeled scenarios. For the modeling part, we choose to extend the existing ADL PCM [43] with confidentiality annotations [46]. Based on the extended metamodel, we then create a PCM model representing the software architecture with confidentiality properties. This extended PCM model is then transformed into a dataflow diagram [48,49]. These dataflow diagrams are then combined with dataflow constraints and analyzed regarding violations of these constraints. We described this analysis method in previous publications [46,47].

4.1 Modeling Confidentiality in PCM

Modeling confidentiality aspects of software architectures in PCM serves two purposes: First, aspects, which are relevant for reasoning about confidentiality are documented. Second, a structured documentation of these aspects enables automated analyses. In the following, we focus on the second purpose since we later on use this for our analysis.

The analysis on dataflow diagrams, which we reuse to analyze software architectures given in PCM, is based on label propagation [49]. To make use of the existing analyses [49], PCM has to provide all information to derive a label propagation graph. In our extended PCM model, we call labels *characteristics*. They are strongly typed and defined in a so-called *data dictionary*. Listing 1.1 illustrates an excerpt of the data dictionary for our running example. We first define

an enumeration `Location` with the literals `EU` and `nonEU`. The same is done for the `Sensitivity` levels. The enumerations define value ranges. Additionally, we specify characteristic types (here `ServerLocation` and `DataSensitivity`), that reference the enumerations and give the values a meaning. For instance, a location can be used as a value for a characteristic, which describes the location of a node or the origin of data, which is allowed to flow to the node. A characteristic always has its type and potentially multiple values.

```
1  enum Location {
2    EU
3    nonEU
4  }
5  enum Sensitivity {
6    Personal
7    Public
8  }
9
10 enumCharacteristicType ServerLocation using Location
11 enumCharacteristicType DataSensitivity using Sensitivity
```

Listing 1.1. Simplified textual representation of a data dictionary

The characteristics for nodes can be added to resources, users and deployed components in PCM. Resources host components, so the characteristics also apply to the components deployed on the resource. Figure 2 illustrates these annotations for our running example by using the PCM tool support. Here we have two servers (`OnPremiseServer`, `CloudServer`). Their characteristics are shown as green labels. Both characteristics refer to the characteristic type `ServerLocation`. One characteristic refers to the value `EU` and the other one refers to the value `nonEU`. These characteristics are transformed to labels of the nodes in the label propagation graph during the analysis.

The characteristics of data flows stem from the characteristics that are assigned to parts of the system. In PCM, data is represented by parameters or return values of services. We refer to both types of data as variables. PCM can apply characteristics to such variables. We extended the means for modeling this application of characteristics to support the characteristic types from data dictionaries: For every variable, a software architect can define multiple assignments. An assignment applies a characteristic, which is specified on the left-hand side of an assignment, if the expression on the right-hand side of an assignment

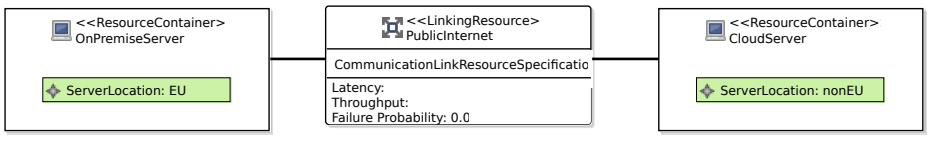

Fig. 2. Resource containers of the running example with location characteristics

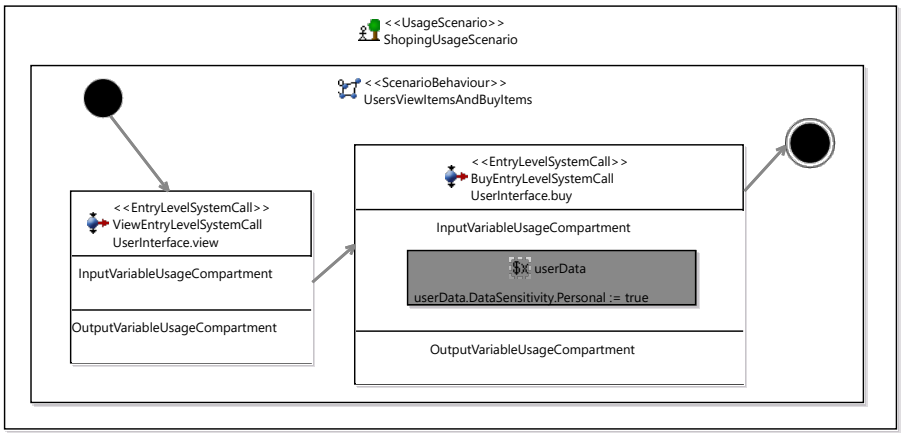

Fig. 3. Usage Scenario of the running example with characteristics assignments

evaluates to true. The right-hand side can be a constant assignment of a truth value, a query for the existence of a characteristic on another variable or a logical connection such as a conjunction of the previously mentioned expressions. It is possible to express a characteristic propagation that uses incoming characteristics to produce outgoing characteristics. In the analysis, these assignments become label propagation functions and the characteristics are treated as labels.

Figure 3 illustrates an assignment in a usage scenario of the PCM tool support. In our running example, there is a service called *buy*. This service has, among other parameters, the parameter `userData`. For this parameter, we would like to specify that the sensitivity is *personal*. We can write this as an assignment of a constant: *userData.DataSensitivity.Personal := true*. This describes that the variable `userData` should be assigned the `DataSensitivity` `Personal`. Describing all details of the expressions and the mapping to dataflow diagrams as part of the analysis are out of scope of the paper. Instead, we refer to previous publications for details on expressions [49] and the mapping [46].

A model to model transformation transforms the extended PCM to a label propagation graph. Figure 4 illustrates a simplified graph based on the running example. We omit all edges that do not represent relevant data for our analysis. Labels on nodes, such as the *EU* label at the actor, define properties of these nodes. Labels on data, such as the *Personal* label at the dataflow originating from the actor, define properties of the exchanged data. Nodes propagate the received labels through outgoing dataflows. In our example, every node is forwarding the data without modification of its properties. The question mark indicates the structural uncertainty introduced by the unknown allocation. After the propagation, every exchanged data has a label. Afterwards, the policy, which in this particular case demands that data with a *Personal* label must never flow to nodes with a *nonEU* label, can be checked.

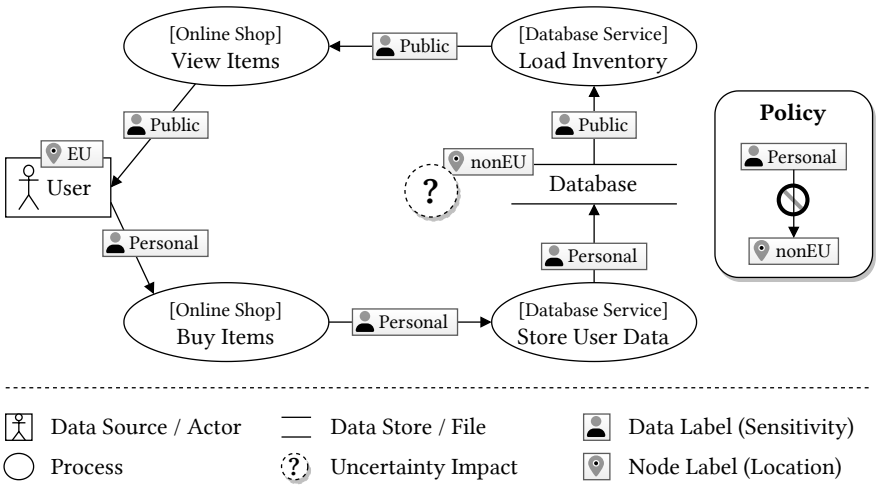

Fig. 4. Simplified label propagation graph under uncertainty of the running example

4.2 Defining and Analyzing Dataflow Constraints

To find confidentiality violations in architectural models, we have to specify prohibited behavior, first. We defined a domain-specific language (DSL) that enables software architects to model dataflow constraints on architectural abstraction [20]. Together with the modeling capabilities using characteristics and assignments, this provides means for architects to analyze confidentiality.

```
1 type DataSensitivity : DataSensitivity
2 type ServerLocation : ServerLocation
3
4 constraint NoPersonalDataToNonEUServers {
5   data.attribute.DataSensitivity.Personal NEVER FLOWS
6   element.property.ServerLocation.nonEU
7 }
```

Listing 1.2. Constraint for prohibiting flows of personal data to non-EU servers

In our running example, a confidentiality violation occurs if personal data flows to components that are located outside the EU. Listing 1.2 shows this constraint formulated using the DSL. We start by reusing the type definitions from the data dictionary. Then, we define the named constraint that shall restrict the flow of personal data to non-EU servers. Each constraint consists of at least one data selector and one element selector. In our example, we select data that has the characteristic value `Personal` of type `DataSensitivity`. We select all elements with the characteristic `ServerLocation` set to `nonEU`. The directive `never flows` implicates that selected dataflows are always forbidden. Thus, whenever a selected dataflow is detected, a violation is automatically reported. The complete analysis process is automated and transparent to software architects [47].

Please note that this only represents a very basic use case of the DSL. For example, we added capabilities to define variables and variable sets that are filled and compared dynamically during the analysis execution. By doing so, architects can describe more complex constraints such as enforcing role-based access control (RBAC) [14]. We also added selectors to reference a specific element of the architectural model or to prohibit any type of dataflow.

The tooling [47] uses Prolog to analyze the architectural model. After the modeling by software architects, both the architectural model and the constraints are automatically transformed to executable Prolog code. Here, the model is transformed into a set of clauses, that represent the dataflows in the system [49]. The dataflow constraint—formulated using the DSL—is transformed into a Prolog query and executed [20]. Besides the DSL, architects can also formulate Prolog queries directly if the DSL is too limiting. Whenever the Prolog engine successfully satisfies the query, a dataflow violation has been found. The results are translated back into the architectural domain and displayed to the architects. By doing so, the violations can be enriched with contextual information such as where the violation occurred, and which characteristics were set. Also, the complete process of transformation of models, constraints, and results as well as the analysis are automated and transparent to the architects.

5 Considering Confidentiality under Uncertainty

With the dataflow analysis from the previous section, we can analyze architectures for confidentiality issues without taking uncertainty into account. As we discussed in previous publications [6,18] the confidentiality of the system can be influenced by uncertainty. This uncertainty can exist in different forms such as in the environment, e.g., no reliable information about the location of users, or the system structure, e.g., the deployment location assumed during the design-time [6]. There exist different means to handle uncertainty such as fuzzy policy evaluation [9] to compensate uncertainty during runtime or probabilistic model solver such as PRISM [23].

In our case, we concentrated on the uncertainty of structural elements such as the deployment or the binding of components. For this type of uncertainty, a possible solution is the creation of different variations without uncertainty. For instance, for our running example, this would involve a different variation of the software architecture for each deployment location. One solution, that uses such variations, is model averaging [39]. Here, multiple variations are created, and afterwards, the results are combined. However, using an average value for confidentiality is not useful since it is a binary decision, whether systems are confidential or not. Therefore, we want to discard solutions that have confidentiality issues. Additionally, creating these different variations manually is cumbersome and error-prone, and analyzing all variations can be very time-consuming. Besides these drawbacks, architects seldom want to optimize regarding only one quality attribute such as confidentiality but want to find the overall best architecture. Therefore, it is beneficial to integrate our dataflow analysis in an existing

architecture design space exploration tool (see Subsection 3.2). In our case, we used PerOpteryx [29], since it already supports an architectural design decision model for modeling uncertain design decisions, the optimization of multiple quality attributes such as costs or performance and is well integrated into the PCM. The first step for integrating our analysis approach is defining a new QML contract for a new quality dimension named confidentiality. These QML contracts define the quality dimensions PerOpteryx [30] can consider. Quality dimensions are the different quality attributes PerOpteryx can optimize. While PerOpteryx already contains a security dimension [8], we decided against it since we wanted to explicitly target confidentiality. The existing security domain is focused more on the costs of introducing security measures and costs of failure.

The confidentiality dimension is modeled as real values where decreasing values are better solutions, since the optimization approach of PerOpteryx currently only support real values. However, we mapped these to a binary classification for the confidentiality analysis results. For each quality dimension, PerOpteryx defines handlers, which can analyze a model for the given quality dimension. Therefore, we defined a new confidentiality handler for the confidentiality domain. This handler acts as an adapter between PerOpteryx and our original dataflow analysis. The main task is to provide the input models for our analysis, run the dataflow analysis and transform the results of our analysis as parameters for the optimization process of PerOpteryx. For the input models, we need to extract the current architecture candidate models. These are the varied architectures models based on the evolutionary search and the design decision model from PerOpteryx. The candidate model of our running example contains one concrete design decision, for instance the deployment of the database on Non-EU servers. This candidate model is now extracted from PerOpteryx and loaded into our analysis. Additionally, the adapter provides the confidentiality requirements for the dataflow analysis. After executing the analysis, we transform the results to the newly added confidentiality dimension. Our analysis can only identify violations based on confidentiality requirements. Currently, these violations are not ranked, and we cannot differentiate which violations are worse since the ranking of confidentiality is not obvious. Even obvious solutions for ranking issues, such as the number of violations, are problematic since scenarios with a higher number of violations might only leak not relevant data or leak data only in very rare cases. At the same time, scenarios with a low number could leak highly sensitive data.

The ranking of violations or quantification of confidentiality can be seen as another research area. Therefore, we choose here a binary classification with the classes *confidential* and *non-confidential*. However, because the confidentiality dimension uses real values, we need to transform the binary classifications. Also, choosing another representation than real values is not possible due to the underlying extension mechanism of PerOpteryx. Also, developing another optimization approach that supports binary classifications would come with additional drawbacks, such as missing integration of other quality metrics. Our adapter transforms confidential results to `-1` and non-confidential results to `1`. Despite the

binary-classification of the dataflow analysis [49] results, the dataflow analysis can support multiple attributes as input values. For instance, in a previous publication [49], we successfully analyzed role-based access control or attribute-based access control approaches. The results then enable the PerOpteryx optimization to rank confidential candidates higher than non-confidential ones. Again, the value of the mapping is partly given by technical restrictions. Since we want to use the restriction option of PerOpteryx to filter/discard non-confidential candidates, the confidential result needs to be lower than the non-confidential one. PerOpteryx currently only supports filtering candidates with too high values. However, the choosing of negative values, does not affect the optimization approach since it explicitly accepts a decreasing integer range as an optimization criterion. Also, the value ∞ is reserved as an error code, as it is usually in the PerOpteryx domain.

In addition to defining the confidentiality dimension and the adapter between PerOpteryx and our dataflow analysis, we extended PerOpteryx by a new contract type for confidentiality. These types are used in the optimization definition of PerOpteryx and define which quality attributes should be optimized. In our case, these would be the confidentiality dimension.

The application of our approach looks like this: First, software architects need to specify the architectural model and extend it with our dataflow extensions (see Subsection 4.1). For the specification, they can user our developed Eclipse tool [47], which provides graphical editors to our models. It extends the existing notions of PCM. Figure 2 illustrate how this extension looks for the resource environment and our running example. Additionally, they need to define the confidentiality constraints either with our DSL (see Subsection 4.2) or use a Prolog constraint. For our running example, the DSL specifications looks like in Listing 1.2. The architectural variations are modeled in the design decision model from Peroperyx. For instance, for our running example, the architect can define here that the database service component can be deployed on two different servers. Besides design decisions, architects need to specify the optimization aspects. Here, PerOpteryx reuses QML contracts [15,36] (see Subsection 3.2). For

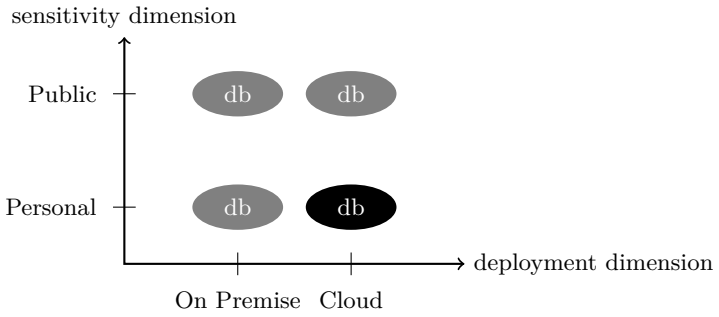

Fig. 5. Overview for possible confidentiality results based on the deployment for the database component and the sensitivity level of the stored data

our running example, the architect creates a new QML contract with the target to achieve `-1` in the confidentiality dimension. This is modeled by an objective with the value `-1` as a goal. This represents a system without confidentiality violations. Afterwards, the architect will get a CSV file with the tested variations and a separate file for the Pareto optimal variation.

The possible results for our running example are illustrated in Figure 5. On the vertical axis, the sensitivity level of the data is shown. In our case, these are `Personal` and `Public`. On the horizontal axis the deployment possibilities, i.e., `On Premise` and `Cloud`, are shown. We added the results for the database service component (`db`) by using different colored ellipses. The grey ones are configurations without confidentiality violations. The black one contains confidentiality violations. We only illustrate the database component since it is in our case the deciding factor. In our running example, we modeled that the database stores `Personal` data. Therefore, our approach returns only the variation with the `Database Service` on premise.

6 Evaluation

In the evaluation, we focus on the combination of PerOpteryx and the dataflow-based confidentiality analysis. The evaluation for PerOpteryx can be found in [30] and for the dataflow-based confidentiality analysis in [48,49]. Our evaluation exploits these results and focuses on the integration of both approaches. Our evaluation follows roughly the Goal Question Metric [3] approach. We first describe our goals, questions, and if applicable the metrics and the study design afterward. Then, we discuss the results, threats to validity, and limitations of the approach.

6.1 Evaluation Goals, Questions, and Metrics

The first goal **G1** is to evaluate the feasibility of our approach. Our evaluation question is **Q1**: *Is it possible to use design space exploration to create different architectural variations which represent the influencing structural uncertainty and required confidentiality?* We do not provide a metric for this question, but we will discuss our experience in our result section.

The second goal **G2** is to evaluate the accuracy of the approach. A high accuracy guarantees that the approach results are meaningful and can be used for design decisions. The first question for the second goal is **Q2.1:** *What is the accuracy of our approach for analyzing confidentiality based on the software architecture?* With this question, we investigate whether the integration will affect the dataflow analysis and might change its result. The second question is **Q2.2**: *What is the accuracy of considering confidentiality results in the software architecture optimization to identify the Pareto optimal candidate?* Here, we investigate whether the results of the confidentiality analysis are correctly interpreted and used for the optimization process. For both questions, we use the metrics precision and recall [53]. For each investigated evaluation scenario,

we create a manual reference output, and this manual reference output is then compared against the automatic result. Each incorrectly classified result is a false positive f_p, and each correctly classified result is a true positive t_p. An expected but missing result is counted a false negative f_n. The accuracy is then calculated by precision $M1.1 = \frac{t_p}{t_p+f_p}$, and recall $M1.2 = \frac{t_p}{t_p+f_n}$.

6.2 Evaluation Design

We perform our evaluation based on four different application examples from the confidentiality [27,32] or design space exploration research community [30].

For answering **Q1**, we combined the input model of PerOpteryx and the dataflow analysis. During this, we checked the model for contradicting model elements. After combining the input models, we run our new analysis and checked whether an error exists.

For answering **Q2.1**, we investigate for each example the generated candidates from PerOpteryx. PerOpteryx also stores the configuration for every candidate and the results of the quality analysis. In our case, this is the classification in confidential and non-confidential. Two authors investigated whether for each candidate the classification is correct. A classification is correct, if the original dataflow [49] analysis would yield the same results also without PerOpteryx. The two authors are not the developers of the original dataflow analysis. Afterwards, we calculated precision and recall.

For answering **Q2.2**. we used the same example as for **Q1**. However, we now looked at the optimization process of PerOpteryx. Here, we compared the suggested Pareto optimal candidates to all generated candidates. Based on this, we calculated precision and recall. If the example also considered other quality attributes as confidentiality, we also included these for our evaluation. If not, we only considered the confidentiality.

Table 1. Original research domains and source for the application examples

Application Examples	Domain	Source
TravelPlanner	Confidentiality	[27,49]
Business Trip Management	Design Space Exploration	[30]
Distance Tracker	Confidentiality	[27,49]
Online Shop	Confidentiality	[32,18]

6.3 Application Examples

Our requirements for our application examples are that they describe the system from an architectural viewpoint. Additionally, the examples need to have at least documentation about either uncertainty and different design decision

or need documentation regarding the confidentiality properties. For application examples missing one of these properties, we added the missing properties manually based on our experience with similar examples and systems. For every example, there is at least one design decision that produces a confidential candidate and one decision for a non-confidential candidate. We have selected our examples from the design space exploration research or confidentiality research domain. Table 1 shows the original research domain for our application examples and their sources.

Our first example is the Travelplanner [27] scenario. It is a confidentiality research scenario and is also used in the evaluation of different similar approaches such as [48,49,56,31]. In our evaluation, we used the version from our ECSA Tutorial [47]. This variant contains 7 components. The scenario describes a flight booking system. A user can book a flight at an airline using their mobile app. The user has to send the credit card data to the airline for booking the flight. However, the credit card data is confidential and should be only accessible by the user. Therefore, the user has to declassify the credit card data manually, and only afterwards the data should be shared with the airline. We extended this basic scenario with a potential design decision that would violate the confidentiality constraint of the credit card data. In our scenario, structural uncertainty exists because it is unknown, where the credit card data should be stored.

Our second example is the Business Trip Management from PerOpteryx [30]. It describes a system to organize and book business trips. For the system, multiple design decisions are modeled. For instance, the deployment of the components can be changed, and there exist multiple implementations of components. However, these different deployment locations and implementations of components also have different costs. Here, we reused the original models from PerOpteryx [30]. It contains 4 components. Since PerOpteryx so far has not considered confidentiality, we extended this example with a confidentiality annotation. For simplicity reasons, we used low and high as values to model the confidentiality. Our confidentiality constraints describe that high data cannot flow to nodes considered low. This is similar to other information flow approaches. In this case, the uncertainty is also the deployment. Additionally, we have a design decision where we exchange some of the implemented components.

The third example is the Distance Tracker [27]. It was already used in the evaluation of our dataflow-based confidentiality analysis [48]. It models a simple mobile application, which tracks the walked distance of the user by a GPS tracker application. The walked distance is then synchronized with a service provider. However, this provider should have no access to the actual GPS data, only to the walked distance. Here, we added a design decision that would let the service provider access the GPS data. The system contains 8 components.

Our fourth example is our running example (see Section 2). We modeled the PCM models and constraints based on the descriptions from [32,18].

The Travelplanner and Distance Tracker [27] are both from the confidentiality research domain. They were both already investigated in previous confidentiality analysis publications [48,49]. Therefore, they fulfill our requirements. The

Business Trip Management [30] is investigated in [30]. It contains an architectural model and multiple design decisions. Therefore, also our requirements are met. The running example has one design decision leading to structural uncertainty and the software architecture is modeled. Therefore, it also fulfills our requirements.

6.4 Results and Discussion

We have investigated each research question for each application example. For **Q1**, we have found no contradicting model elements. We could model every aspect necessary for the example. Additionally, we executed every example and looked for errors in the generated results. PerOpteryx marks each candidate with an error during the analysis with an ∞ value. Our results did not contain any ∞ values, therefore we assume, that no significant errors happened. This finding indicates that a combination is technically feasible.

For every question of **G2** (**Q2.1** and **Q2.2**), we get a precision and a recall of 1.0, which are the perfect results. The results are good because our example models are small, which simplifies the output. Also, most of the decisions are easily understandable and can be deduced easily without automatic tooling. However, these smaller scenarios are better to identify implementation problems or other errors easily. Additionally, in more complex scenarios, deducing the confidentiality is not that easy, especially if there are multiple different deployment decisions for components. Table 2 illustrates the results of our analysis. It shows an excerpt of the analyzed candidate models for the Business Trip Management. The first column is the cost. The second column are the confidentiality results (with -1 for confidential and +1 for non-confidential). The next three columns describe the deployment locations for components. The last column describes the used implementation (the normal implementation (BookingSystem) or a faster one (QuickBooking)). For brevity reasons, we only show an excerpt of the results. In this configuration the Server1 and Server3 are considered as High and Server2 as low. Based on these values we calculated precision and recall.

For **Q2.1**, the results indicate that using the architectural candidates from PerOpteryx as variations for the confidentiality analysis will not affect the confidentiality analysis. The results for our examples are the same as if we would have manually created model variations based on the uncertainty and afterwards analyzed them with our dataflow analysis [49]. For **Q2.2**, the results indicate that the optimization process of PerOpteryx can successfully consider the confidentiality results. This could enable PerOpteryx to optimize the architecture for confidential systems and even discard non-confidential architectural variations.

In summary, these results could enable architects to optimize software architectures regarding different quality metrics including confidentiality. This is indicated by our results for **G1** and **G2**. **G1** demonstrates that both analyses can be combined into one working analysis for our examples. So far there are no conceptual problems known. The results for **G2** indicate, that the found architectural candidates can be successfully analyzed regarding their confidentiality. Additionally, it indicates that the optimization process potentially can

use the result from the confidentiality analysis. Therefore, our approach could optimize for confidential software architectures. Also, our approach should be less error-prone than creating manually the variations. At the manual creation of variations, architects need to manually track the different variations. In contrast, in our analysis architects need only to specify the variation points and the analysis automatically generates architecture candidates. In addition, since we used PerOpteryx which supports optimization for many other quality attributes, architects could get Pareto optimal candidates for the software architecture.

Table 2. Excerpt of results for candidates of the Business Trip Management example

Cost	Confidentiality	C1	C2	C3	BookingSystem
5734.33	-1	Server3	Server3	Server2	QuickBooking
2749.12	+1	Server2	Server2	Server3	BookingSystem
3924.26	-1	Server1	Server3	Server2	BookingSystem
3497.23	-1	Server3	Server3	Server2	BookingSystem
1570.53	-1	Server3	Server3	Server1	BookingSystem
1647.13	-1	Server3	Server2	Server3	BookingSystem
7609.77	+1	Server1	Server2	Server3	QuickBooking
			...		

6.5 Threats to validity

We categories our threats to validity based on the guidelines from Runeson and Höst [44].

Internal Validity describes that only investigated factors affect the result. The internal validity is threatened by the manual creation of the reference output and the extension of the existing examples. To reduce the risk of the reference output, we compared the reference output with the output of the dataflow analysis [49] without our extension. Additionally, the small size helps to understand the results. To reduce the second risk, we added the missing properties in the examples based on our experience with similar examples, where the missing properties are modeled. Another threat to internal validity is that we only investigated the adapter and not the optimization process of PerOpteryx or the accuracy of the used confidentiality analysis. However, we integrated our integration by using the dedicated extension mechanism in PerOpteryx and only adapted the user interface. For the dataflow analysis [49], we only added an additional output method. Otherwise, we have not changed the source code. Therefore, we consider this threat to be low since both approaches are well established and are thoroughly evaluated in different publications such as [48,49,46,29,8].

External Validity describes how transferable the results are for others. The usage of application examples might provide better insights but might affect the generalizability of the results. Therefore, we chose mostly external examples from different research domains. Our first and third examples are from the confidentiality domain and the second one is originally an optimization example for performance and cost. Using different examples lowers the risk. Nevertheless, in the future, we want to investigate real application case studies and not only research examples.

Construct Validity describes whether the intended properties are relevant for the goal. Here the properties are the metrics precision and recall. These metrics are also used in the evaluation of the original confidentiality analysis [49]. Using the same metrics lowers the risk. We did not investigate the scalability of the approach. However, since the added adapter contains not too much complex logic the scalability should be similar to the foundational approaches.

Reliability is how reproducible the results are for other researchers. Using statistical metrics for G2 reduces the subjectivity of the result and increases reproducibility. Konersmann et al. [28] state the lack of reproduction packages in current software engineering research. Thus, we provide a dataset [55] containing all the relevant models and the analysis. Furthermore, the extension is published as an open-source tool[3] as well as all the necessary dependencies. This allows other researchers to reproduce the result more easily and increases the reuse.

6.6 Limitations

One limitation is currently the restricted variation creation based on the architectural decision model of PerOpteryx. While this already allows modeling structural uncertainty such as different deployments, other aspects like different variations of user roles are not possible. However, this does not mean, that our confidentiality analysis [49] approach cannot handle different roles. In fact, it can use different roles or even arbitrary attributes like in Attribute-based access control and we already demonstrated it in [49]. Only it does not support uncertainty in these attributes, yet. A possible solution might be the extension or the transformation of the design decision model to an uncertainty variation model. This would also help to create an impact analysis that identifies affected elements of uncertainty. Another limitation, so far, is the missing differentiation of confidentiality issues. Currently, the optimization of PerOpteryx only binary classifies the results and optimize to a system without confidentiality violations. However, it might be beneficial to allow certain types of violations to get better overall system performance. This is similar to Bures et al. [7], where access control policies are adapted and sometimes lessened to guarantee a working system. Another first approach for quantifying security within PCM is presented by Reiche et al. [42]. Another problem is due to the design-time nature of our

[3] `https://github.com/FluidTrust/Palladio-Addons-DataFlowConfidentiality-DSE`

approach. Our model might differ from the actually implemented system, and therefore, the result of the analysis might not be true for the actual system. However, approaches like [57,35] try to tangle this issue. Additionally, we do not consider an explicit attacker model. Therefore, we cannot consider different attacker behaviors in the optimization process. A solution might be the integration of our attacker analysis [56].

7 Related Work

We split our related work in design space exploration, model-driven confidentiality analysis, and uncertainty.

7.1 Design Space Exploration

Sobhy et al. [50] present an overview of different design space exploration considering uncertainty. In it, PerOpteryx [30] is categorized as a search-based approach. Another design space exploration tool is ArcheOpterix [1]. It can also optimize a given architecture for multiple criteria. However, it does not support a confidentiality analysis, and the design space modeling is more restrictive. GuideArch [12] uses fuzzy logic to handle uncertainty in the design phase. This uncertainty can be reduced during the development and, therefore, GuideArch might provide more precise results. However, currently, there is no support for confidentiality. Last, there is already a PerOpteryx extension for security [8]. In contrast to our approach, they modeled these by a concept of concerns. These concerns describe which design decisions are dependent on each other. In our analysis we focus on the direct impact on confidentiality.

7.2 Model-driven Confidentiality Analysis

UMLsec [25] extends UML by defining a security profile. It supports different kinds of analyses, such as information flow or access control. In contrast to our confidentiality analysis, it does not support access control on data, and there is no support to handle structural uncertainty. Another UML security profile is SecureUML [33]. It supports role-based access control together with OCL statements to support dynamic properties. However, the analysis does also not consider structural uncertainty. SecDFD [52] is also a dataflow analysis approach. They do not support custom analysis definitions such as we with our DSL (see Subsection 4.2) or the considering of structural uncertainty with our architectural design decision model from PerOpteryx. Gerking et al. [17] and the iFlow [26] approach are also information flow analyses. The first one targets especially Cyber-physical systems and keeping the real-time properties. It again does not support structural uncertainty. The same applies to the iFlow approach. We also developed another confidentiality analysis for software architectures [56]. However, there the focus is on the architectural propagation of attackers and not dataflow or uncertainty.

7.3 Uncertainty

Garlan [16] discusses why it is necessary to consider uncertainty during the software engineering. Uncertainty itself is in general classified in Walker et al. [54] without special relation to software engineering. Perez-Palacin and Mirandola [39] extended this classification and combined it with other classifications, such as [2]. They also discuss mechanisms to handle different types of uncertainty. However, they do not consider confidentiality. Additionally, they focus on self-adaptive systems and do not specifically aim at handling uncertainty on architectural abstraction.

Esfahani and Malek [11] also discuss uncertainty in the context of self-adaptive systems. They highlight the problem of uncertainty in the environment of a software system, e.g., due to the deployment. Ramirez et al. [41] present different sources of uncertainty and a scheme to describe uncertainty in different development phases. Regarding design-time confidentiality, they lack in precision and name related uncertainty as *inadequate design* or *unverified design*. Still, they also list *unexplored alternatives* and *misinformed trade-off analysis* which motivates our work. Troya et al. [51] provide a current survey of different approaches regarding confidentiality. They provide another classification of uncertainty and point out the growth of research with relation to uncertainty in software models. However, they also mention the lack of repeatable results as well as tool support which we tried to also address with our work.

Mahdavi et al. [22] investigated the research community of self-adaptive systems about uncertainty. They provide an overview of the current state of the community. They find that the current state of the art lacks in including non-functional requirements both as optimization goal as well as side effects. By including confidentiality in the general PerOpteryx analysis, we can optimize towards confidentiality but also detect violations. Thus, our approach addresses this issue. We also at least partially address the mentioned problem of dealing with concurrent sources of uncertainty, at least regarding structural uncertainty.

The Design-Time Uncertainty Management (DeTUM) [13] is a tool to handle uncertainty during the design-time. It introduces uncertainty in the start phase and then resolves it in later stages. However, the authors do not mention security-related quality attributes like confidentiality. Lytra and Zdun [34] propose the use of fuzzy logic to incorporate inherent uncertainty into reusable, architectural design decisions. This shall enable software architects to share and reuse knowledge about the impact of uncertainty on quality attributes. Although this approach can also handle security-related quality attributes, violations due to integration issues remain hidden. Here, the integration of a dedicated analysis—such as our confidentiality analysis—helps identify also fine-grained problems.

Boltz et al. [5] present an approach to extend our dataflow-based confidentiality analysis to handle environmental uncertainty. They use fuzzy inference systems to calculate the impact of environmental variables on confidentiality-related attributes. Although the there discussed environmental uncertainty is another instance of known uncertainty, statements of its influence are not directly comparable to statements about an architecture's structure under uncertainty.

In summary, there exist many approaches to categorize uncertainty and some also provide approaches to handle certain uncertainty types. However, they have so far not considered confidentiality explicitly. Also, there is a focus on self-adaptive systems with a lack of systematic approaches on architectural abstraction [22].

8 Conclusion

We have presented an approach to consider structural uncertainty and confidentiality in software architectures. For this, we combined our dataflow-based confidentiality [48,49,47] with an architectural optimization approach [30]. Besides considering confidentiality, our approach can be used to determine Pareto optimal software architectures regarding multiple quality attributes. We demonstrated that we could detect confidentiality violations for our examples, and these exemplary confidentiality violations are correctly considered in the optimization process. This indicates that our approach could work correctly and might help architects to determine the Pareto optimal architecture candidate.

In the future, we want to extend the considered uncertainties and further investigate the accuracy with real-world case studies. We also aim to better understand the impact of different types of uncertainty on confidentiality and create a classification scheme. This shall help in discussing the impact of uncertainty and find appropriate mitigation strategies. Also, we want to extend the presented approach of handling uncertainty by analyzing different architecture variations. Last, we want to repeat the survey regarding the applicability and usability of our dataflow approach, which we conducted during a tutorial. The future study should include our new extension and address more participants.

Acknowledgement

We like to thank Oliver Liu, who helped in developing this approach during his Bachelor thesis.

References

1. Aleti, A., Bjornander, S., Grunske, L., Meedeniya, I.: ArcheOpterix: An extendable tool for architecture optimization of AADL models. In: ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software. pp. 61–71 (2009). https://doi.org/10.1109/MOMPES.2009.5069138
2. Armour, P.G.: The five orders of ignorance. Commun. ACM **43**(10), 17–20 (oct 2000). https://doi.org/10.1145/352183.352194
3. Basili, G., Caldiera, V.R., Rombach, H.D.: The goal question metric approach. Encyclopedia of software engineering pp. 528–532 (1994)
4. Boehm, B., Basili, V.: Software defect reduction top 10 list. Computer **34**(1), 135–137 (Jan 2001). https://doi.org/10.1109/2.962984

5. Boltz, N., Hahner, S., Walter, M., Seifermann, S., Heinrich, R., Bures, T., Hnetynka, P.: Handling environmental uncertainty in design time access control analysis. In: 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE (2022), accepted, to appear
6. Bures, T., Hnetynka, P., Heinrich, R., Seifermann, S., Walter, M.: Capturing dynamicity and uncertainty in security and trust via situational patterns. In: ISoLA 2020. Lecture notes in computer science (LNCS), vol. 12477, pp. 295–310. Springer Verlag (2020). https://doi.org/10.1007/978-3-030-61470-6_18
7. Bureš, T., Gerostathopoulos, I., Hnětynka, P., Seifermann, S., Walter, M., Heinrich, R.: Aspect-oriented adaptation of access control rules. In: 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 363–370 (2021). https://doi.org/10.1109/SEAA53835.2021.00054
8. Busch, A., Schneider, Y., Koziolek, A., Rostami, K., Kienzle, J.: Modelling the structure of reusable solutions for architecture-based quality evaluation. In: 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). pp. 521–526 (2016). https://doi.org/10.1109/CloudCom.2016.0091
9. Casola, V., Preziosi, R., Rak, M., Troiano, L.: A reference model for security level evaluation: Policy and fuzzy techniques. J. Univers. Comput. Sci. **11**(1), 150–174 (2005)
10. Council of European Union: REGULATION (EU) 2016/679 (general data protection regulation), https://eur-lex.europa.eu/eli/reg/2016/679/2016-05-04
11. Esfahani, N., Malek, S.: Uncertainty in Self-Adaptive Software Systems, p. 214–238. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35813-5_9
12. Esfahani, N., Malek, S., Razavi, K.: GuideArch: Guiding the exploration of architectural solution space under uncertainty. In: 2013 35th International Conference on Software Engineering (ICSE). pp. 43–52. IEEE (2013). https://doi.org/10.1109/ICSE.2013.6606550, http://ieeexplore.ieee.org/document/6606550/
13. Famelis, M., Chechik, M.: Managing design-time uncertainty. In: MODELS. p. 179. IEEE Press (2017). https://doi.org/10.1109/MODELS.2017.24
14. Ferraiolo, D., Cugini, J., Kuhn, D.R.: Role-based access control (RBAC): Features and motivations. In: ACSAC'95. pp. 241–248 (1995)
15. Frolund, S., Koistinen, J.: A language for quality of service specification. Tech. rep., HP Labs Technical Report, California, USA (1998)
16. Garlan, D.: Software engineering in an uncertain world. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research. p. 125–128. FoSER '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1882362.1882389
17. Gerking, C., Schubert, D.: Component-Based Refinement and Verification of Information-Flow Security Policies for Cyber-Physical Microservice Architectures. In: ICSA'19. pp. 61–70. IEEE (mar 2019). https://doi.org/10.1109/ICSA.2019.00015, https://ieeexplore.ieee.org/document/8703909/
18. Hahner, S.: Architectural access control policy refinement and verification under uncertainty. In: Companion Proceedings of the 15th European Conference on Software Architecture. CEUR Workshop Proceedings, vol. 2978. RWTH Aachen (2021), 46.23.03; LK 01
19. Hahner, S.: Dealing with uncertainty in architectural confidentiality analysis. In: Proceedings of the Software Engineering 2021 Satellite Events. p. 1–6. Gesellschaft für Informatik, Virtual (2021)

20. Hahner, S., Seifermann, S., Heinrich, R., Walter, M., Bures, T., Hnetynka, P.: Modeling data flow constraints for design-time confidentiality analyses. In: 2021 IEEE International Conference on Software Architecture Companion (ICSA-C). pp. 15–21. IEEE (2021). https://doi.org/10.1109/ICSA-C52384.2021.00009
21. Heinrich, R., Seifermann, S., Walter, M., Hahner, S., Reussner, R., Bureš, T., Hnětynka, P., Pacovský, J.: Dynamic access control in industry 4.0 systems. In: Digital Transformation, chap. 6. Springer (2022), accepted, to appear
22. Hezavehi, S.M., Weyns, D., Avgeriou, P., Calinescu, R., Mirandola, R., Perez-Palacin, D.: Uncertainty in self-adaptive systems: A research community perspective. ACM Trans. Auton. Adapt. Syst. **15**(4) (dec 2021). https://doi.org/10.1145/3487921
23. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: Prism: A tool for automatic verification of probabilistic systems. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 441–444. Springer (2006)
24. ISO Central Secretary: Information technology — security techniques — information security management systems — overview and vocabulary. Standard ISO/IEC 27000:2018, International Organization for Standardization, Geneva, CH (2018), https://www.iso.org/standard/73906.html
25. Jürjens, J.: UMLsec: Extending UML for Secure Systems Development, vol. 2460, p. 412–425. Springer Berlin Heidelberg (2002). https://doi.org/10.1007/3-540-45800-X_32
26. Katkalov, K., Stenzel, K., Borek, M., Reif, W.: Model-driven development of information flow-secure systems with iflow. In: SOCIALCOM. pp. 51–56 (2013). https://doi.org/10.1109/SocialCom.2013.14
27. Katkalov, K.: Ein modellgetriebener Ansatz zur Entwicklung informationsflusssicherer Systeme. doctoralthesis, Universität Augsburg (2017)
28. Konersmann, M., et al.: Evaluation methods and replicability of software architecture research objects. In: ICSA. IEEE (2022)
29. Koziolek, A.: Automated Improvement of Software Architecture Models for Performance and Other Quality Attributes. Ph.D. thesis, Karlsruher Institut für Technologie (KIT) (2011). https://doi.org/10.5445/IR/1000024955
30. Koziolek, A., Koziolek, H., Reussner, R.: Peropteryx: automated application of tactics in multi-objective software architecture optimization. In: Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems – ISARCS. pp. 33–42 (2011)
31. Kramer, M., Hecker, M., Greiner, S., Bao, K., Yurchenko, K.: Model-driven specification and analysis of confidentiality in component-based systems. Tech. Rep. 12, KIT-Department of Informatics (2017). https://doi.org/10.5445/IR/1000076957
32. Liu, O.: Design space evaluation for confidentiality under architectural uncertainty (2021). https://doi.org/10.5445/IR/1000139590
33. Lodderstedt, T., Basin, D., Doser, J.: Secureuml: A uml-based modeling language for model-driven security. In: «UML» 2002 — The Unified Modeling Language. vol. 24, p. 426–441. Springer, Berlin, Heidelberg (2002), http://link.springer.com/10.1007/3-540-45800-X_33
34. Lytra, I., Zdun, U.: 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. p. 43–46. SESoS '13, Association for Computing Machinery (Jul 2013). https://doi.org/10.1145/2489850.2489859

35. Monschein, D., Mazkatli, M., Heinrich, R., Koziolek, A.: Enabling consistency between software artefacts for software adaption and evolution. In: ICSA. pp. 1–12 (2021). https://doi.org/10.1109/ICSA51549.2021.00009
36. Noorshams, Q., Martens, A., Reussner, R.: Using quality of service bounds for effective multi-objective software architecture optimization. In: Proceedings of the 2nd International Workshop on the Quality of Service-Oriented Software Systems. QUASOSS '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1858263.1858265
37. OWASP: A04:2021 – insecure design, https://owasp.org/Top10/A04_2021-Insecure_Design/
38. OWASP: Top ten web application security risks, https://owasp.org/www-project-top-ten/
39. Perez-Palacin, D., Mirandola, R.: 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. p. 3–14. ICPE '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2568088.2568095
40. Piper, D.: Dla piper gdpr fines and data breach survey: January 2022, https://www.dlapiper.com/de/germany/insights/publications/2022/1/dla-piper-gdpr-fines-and-data-breach-survey-2022/
41. Ramirez, A.J., Jensen, A.C., Cheng, B.H.C.: A taxonomy of uncertainty for dynamically adaptive systems. In: 2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). p. 99–108 (Jun 2012). https://doi.org/10.1109/SEAMS.2012.6224396
42. Reiche, F., Schiffl, J., Weigl, A., Heinrich, R., Beckert, B., Reussner, R.: Model-driven quantification of correctness with palladio and key. Tech. rep., Karlsruher Institut für Technologie (KIT) (2021). https://doi.org/10.5445/IR/1000128855
43. Reussner, R., Becker, S., Happe, J., Heinrich, R., Koziolek, A., Koziolek, H., Kramer, M., Krogmann, K.: Modeling and Simulating Software Architectures – The Palladio Approach. MIT Press, Cambridge, MA (10 2016), http://mitpress.mit.edu/books/modeling-and-simulating-software-architectures
44. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering **14**(2), 131 (2008). https://doi.org/10.1007/s10664-008-9102-8
45. Schulz, S., Reiche, F., Hahner, S., Schiffl, J.: Continuous secure software development and analysis. In: Proceedings of Symposium on Software Performance 2021. Leipzig, Germany (Nov 2021)
46. Seifermann, S., Heinrich, R., Werle, D., Reussner, R.: A unified model to detect information flow and access control violations in software architectures. In: Proceedings of the 18th International Conference on Security and Cryptography, SECRYPT 2021, Virtual, Online, 6 July 2021 - 8 July 2021. pp. 26–37. SciTePress (2021). https://doi.org/10.5220/0010515300260037
47. Seifermann, S., Walter, M., Hahner, S., Heinrich, R., Reussner, R.: Identifying confidentiality violations in architectural design using palladio. In: ECSA-C 2021. vol. 2978. CEUR-WS.org (2021), 46.23.03; LK 01
48. Seifermann, S., Heinrich, R., Reussner, R.: Data-driven software architecture for analyzing confidentiality. In: ICSA. pp. 1–10. IEEE (2019). https://doi.org/10.1109/ICSA.2019.00009, https://ieeexplore.ieee.org/document/8703910/
49. Seifermann, S., Heinrich, R., Werle, D., Reussner, R.: Detecting violations of access control and information flow policies in data flow diagrams. JSS (2021)

50. Sobhy, D., Bahsoon, R., Minku, L., Kazman, R.: Evaluation of Software Architectures under Uncertainty: A Systematic Literature Review. ACM Transactions on Software Engineering and Methodology **1**(1), 50 (2021)
51. Troya, J., Moreno, N., Bertoa, M., Vallecillo, A.: Uncertainty representation in software models: a survey. Software and Systems Modeling **20** (08 2021). https://doi.org/10.1007/s10270-020-00842-1
52. Tuma, K., Scandariato, R., Balliu, M.: Flaws in flows: Unveiling design flaws via information flow analysis. In: ICSA. p. 191–200 (2019). https://doi.org/10.1109/ICSA.2019.00028
53. Van Rijsbergen, C., Van Rijsbergen, C.: Information Retrieval. Butterworths (1979)
54. Walker, W., Harremoës, P., Rotmans, J., Sluijs, J., Asselt, M., Janssen, P., Kraus, M.: Defining uncertainty: A conceptual basis for uncertainty management in model-based decision support. Integrated Assessment **4** (03 2003). https://doi.org/10.1076/iaij.4.1.5.16466
55. Walter, M., Hahner, S., Seifermann, S., Bures, T., Hnetynka, P., Pacovský, J., Heinrich, R.: Dataset: Architectural optimization for confidentiality under structural uncertainty, `https://doi.org/10.5281/zenodo.6569353`
56. Walter, M., Heinrich, R., Reussner, R.: Architectural attack propagation analysis for identifying confidentiality issues. In: ICSA (2022)
57. Yurchenko, K., et al.: Architecture-driven reduction of specification overhead for verifying confidentiality in component-based software systems. In: MODELS (Satellite Events). pp. 321–323 (2017)