Context-Based Confidentiality Analysis for Industrial IoT

Nicolas Boltz, Maximilian Walter and Robert Heinrich
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
Email: nicolas.boltz@student.kit.edu, maximilian.walter@kit.edu, robert.heinrich@kit.edu

***Abstract*—In this research paper, we present an approach for an analysis process, which can find confidentiality issues in data-exchange, on the architectural level of Industrial Internet of Things software systems. Existing approaches provide an insufficient definition of dataflow or lack support of finely granulated information for providing confidentiality. Based on an existing modeling and analysis process for Data-Driven Software Architecture, we extend the role-based approach for access control, with a model to model transformation utilizing a context-based approach. Using a case study based evaluation we show that our approach works accurately and scales in a way that is feasible for big organizations.**

***Index Terms*—industrial IoT, software architecture, dataflow, confidentiality**

I. Introduction

Boyes et al. [1] define the Industrial Internet of Things (IIoT) as a system comprising networked smart objects, cyber-physical assets, associated generic information technologies, and optional cloud or edge computing platforms. Combined, the system enables real-time, intelligent, and autonomous access, collection, analysis, communications, and exchange of process, product and/or service information, within the industrial environment, so as to optimize overall production value. Organizations benefit from this value by being able to implement self-organizing productions, and supply chains, with ad-hoc cooperation between the machines, humans, and organizations involved in the production process. This may improve product or service delivery, boost overall productivity, reduce labor cost and energy consumption, as well as offering small-batch, and/or customized products more cost-efficiently. Entities in these systems represent humans and machines, or suppliers, producers, and processes. Humans communicate with the system by using mobile end-user terminals, like smartphones or notebooks. IIoT systems use and process the data of these entities as decentralized resources that communicate with each other. Since in IIoT, the production process is very dynamic, contexts of all participating entities often change. As a lot of data exchange can happen with outside organizations, like, for example, suppliers or producers, there is much concern about unauthorized access to each other's data. The high flexibility, complexity and amount of data-exchange, makes it more critical than ever to be able to ensure the confidentiality and security of data [2].

An analysis process that can find confidentiality issues within the system is needed. The architectural point of view allows for easier modeling and analyzing software architecture regarding performance prediction. However, to analyze confidentiality, the flow and processing of data in the software system are crucial. While there are existing approaches to dataflow modeling on the architectural level, they lack the support for finely granulated information to provide confidentiality.

Our approach introduces a model for providing context-based confidentiality in dataflow modeling on the architectural level. The model introduces a basic set of contexts as a way to represent properties of active system users (subjects), as well as a way of adding information of the system's or organization's environment to the software architecture. We extend the role-based approach for access control of an existing modeling approach for Data-Driven Software Architecture (DDSA) [3] with our context-based approach. To integrate the context-based information to the dataflow analysis of the DDSA approach, we introduce a model to model transformation. The transformation converts context-based access control data to information which can be regarded by the DDSA analysis process. We assume that our extension of the DDSA process improves usability and reduces error-proneness in creating context-dependent access control policies when compared to the original approach for DDSA. Knowledge about the meaning of each role could only be archived by either using naming conventions or a manually managed table, mapping roles to a written description. Changes to the access rights might lead to inconsistencies or errors throughout the modeled system. Analyzing a system with inconsistencies regarding its access rights can yield invalid results, which are hard to detect. The additional step we add to the DDSA process automates the creation of the DDSA model elements used for access control, removing the error-proneness. Additionally, we believe the context-based extension provides a more natural way of creating access control policies by defining the basic properties of the system's subjects.

For evaluation, we applied the transformation of our approach to 17 scenarios. The scenarios represent equivalence classes of possible context combinations to evaluate the accuracy of the transformation. Models with an increasing number of various context-based information are used to evaluate the scalability of our transformation. The base for the evaluation scenarios and models are the same two case studies that have already been used to evaluate the DDSA [3] approach. A version of the context-based meta-model, as well as the transformation

and evaluation source code has been made publicly available in a data set[1].

The remainder of this paper is structured as follows: We discuss the state of the art in Section 2. In Section 3, we explain Palladio and the DDSA process as foundations. Section 4 presents an overview of our approach, as well as a running example throughout the transformation process. Section 5 covers the evaluation of the transformation. Section 6 concludes the paper.

II. Foundation

Our approach relies on two main foundations, Palladio [4] as a base for designing software architectures (SA) and the Data-Driven Software Architecture (DDSA) [3], for a dataflow definition and analysis on the architectural level.

Palladio is a tool-supported software architecture simulation approach, that is used to predict an architectures Quality of Software properties, like performance or reliability.

The Palladio Component Model (PCM) is a detailed metamodel of component-based software architectures [5]. The basic PCM is made up of a repository, system, resource environment, allocation, and usage metamodel, each representing a different architectural view on a system. The repository model describes components and their interfaces, with the inner behavior of components being described by service effect specifications (SEFF). The system model combines components that are described in the repository model to specify the software architecture. The resource environment model describes the specifications of available processing resources. The allocation model describes which components are deployed on which resource. The usage model describes the behavior of users interacting with the system [4].

Data-Driven Software Architecture: To provide a software architecture description, with a concept of dataflow, we use an extension of the PCM, called Data-Driven Palladio. This extension, together with an analysis process for confidentiality, make up a development process of, so-called, Data-Driven Software Architectures [3].

The Data-Driven Extension introduces data and data processing operators as first-class entities. Data-Driven Palladio supplies a metamodel, which defines how data is represented and what kind of operations can be performed on data. Each data can have sets of characteristics, which represent abstract meta-data of the affiliated data [6]. The process, as shown in Fig. 4, starts with defining and extending a software architecture. The analysis is realized as queries to a Prolog program that a transformation chain derives from the DDSA. The confidentiality analysis uses a Role-Based Access Control (RBAC) strategy to detect access right mismatches [3].

III. State of the Art

There is a wide range of approaches addressing confidentiality. We split these approaches in ones which focus on access control strategies to achieve confidentiality and ones that focus on the mostly model-based development of secure systems.

Access control strategies are used to selectively restrict access to a resource if a user is not able to satisfy a predefined set of access rules. Applying these rules to a dataflow provides confidentiality of the accessed data. Kalam et al. [7] describe the Organization Based Access Control (OrBAC) model, which extends RBAC [8] by introducing the concept of organizations and contexts. Organizations are a set of subjects, which can be users, or other organizations, thereby forming a hierarchy. F. Cuppens and A. Miège [9] have extended OrBAC by modeling explicit contexts. OrBAC offers an implementation of an OrBAC API and a policy editor with a GUI, called MotOrBAC [10]. Attribute-Based Access Control (ABAC), as described by V.C. Hu et al. [11], defines a way of access control using policies. Policies combine attributes and can either be access-granting or access-denying. Policies can work together, build upon, or overwrite each other, thereby building a complex boolean rule set. Attributes can be either atomic-valued, only containing a single defining atomic value, or set-valued, containing multiple atomic values to define the attribute. Their attributes can be compared to static values or each other, to evaluate policies.For the development of secure systems, we focus on approaches that use structured system descriptions to carry out automated analyses as described in the survey of Nguyen et al. [12]. Therefore, we do not consider broadly applicable approaches like Threat Modeling [13], as one of our goals is to provide a more guided way of ensuring confidentiality. UMLSec [14] defines a UML profile for specifying security properties of systems. Even though systems can be verified by the CARiSMA tool [15], UMLSec only supports RBAC [8] for defining access rules. Additionally, different from the DDSA approach (Sec. II) that works on the data path, UMLSec only works on the control path, which makes the dynamic formulation of constraints difficult. The SecureUML [16] approach is set up similarly to UMLSec. It extends UML diagrams with roles, permissions, and users, as well as access control policy annotations for UML elements. Similar to UMLSec, the SecureUML approach is based on the RBAC strategy. Even though the RBAC strategy has been extended by SecureUML, it cannot be easily extended to provide context-based confidentiality. The iFlow approach [17] defines another UML profile and the MODELFLOW language, which can be used to model systems, including their behavior and security requirements. The information flow analysis considers security domains that can be mapped to access control. But setting up such security domains and transitions in between domains, on the granularity of methods can be challenging for users. The SecDFD approach of K. Tuma, R. Scandariato, and M. Balliu [18] defines a data flow diagram that has been enriched with security concepts and an analysis process that verifies the correctness of the concepts used. The approach is implemented as a publicly available Eclipse plugin and is supported by a Domain-Specific Language. For large systems, like in the field of IIoT, describing the dataflow, as well as the necessary software architecture elements, in a single SecDFD can be difficult and obscure for users. Additionally, if an

[1]https://doi.org/10.5281/zenodo.3732448

architecture description of a system already exists, parts of the architectural information need to be duplicated to realize a SecDFD. This brings up the need and error-proneness of keeping them consistent for each change in the architecture. Although runtime is not in the scope of this work, we can exploit our iObserve approach [19] in order to update design time models by observations during runtime to make use of the analysis process proposed in this paper based on architectural runtime models.

IV. Modeling Contexts for Assuring Confidentiality

With our first contribution, we define a framework that represents context-based confidentiality properties in IIoT systems in the form of an architectural meta-model of an IIoT system. The architectural model extension is comprised of two meta-models, the *SubjectModel* and *ContextModel*, and is based on the PCM (Sec. II). The structure and contents of the *SubjectModel* as well as the types of contexts modelled in the *ContextModel*, have been abstracted and derived from workshops with industrial participants [20], [21]. The *SubjectModel*

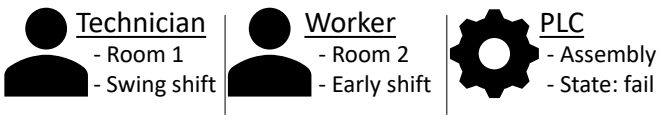

Fig. 1: Subjects of running example.

defines subjects that actively communicate via the IIoT system. *Subjects* can either be human users, organizations, or non-human resources like machines. Organizations may represent a company, but can also be used to represent organizational units. Organizations and organizational units form hierarchical trees. Child nodes represent organizations that are owned by the organization, represented by the parent node. Additionally, organizations can own users and resources, representing that they are associated with the organization.

The *ContextModel* predefines certain types of contexts, which

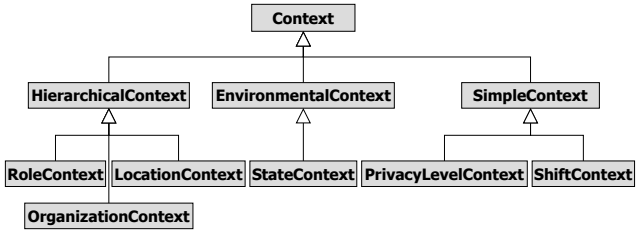

Fig. 2: Class diagram excerpt of *ContextModel*.

represent properties. The contexts can be split up into three types according to their underlying properties. An excerpt showcasing the class structure of the *ContextModel* and all context classes created for this paper are shown in Fig. 2.

Simple contexts represent a property based on a fixed global value. An example is an affinity to a certain shift or privacy level, where each shift or privacy level is represented by a single enum literal.

Environment contexts represent a property that is dependent on the property of another entity of the modeled system. These contexts are mainly used for defining policies. *StateContexts*, for example, reference an entity of the *SubjectModel* which can hold a state property. Additionally, it defines a target state, that can be matched with the state of the entity. For our example in Fig. 1 we have created a resource which represents a programmable logic controller (PLC). These resources can hold a state property, which indicates whether the resource is running or in a fail state. As shown in Fig. 3c we have also created the policy 'PLC log access policy', which contains a StateContext and is meant to regulate access to the log files of the PLC. This StateContext references the PLC resource and has a defined target state of 'fail', meaning that the log files of the PLC should only be able to be accessed when the PLC is in a fail state and requires technical support.

Hierarchical contexts represent properties that have hierarchical dependencies. An example is the current location of a worker within a production plant. A production plant may contain buildings which are made up of areas, which in turn contain rooms. This make-up can be schematically represented by a tree, as shown in the example in Fig. 3a, where child nodes represent a more precise property than their parent nodes.

Sets of multiple contexts are used in two ways. One is to

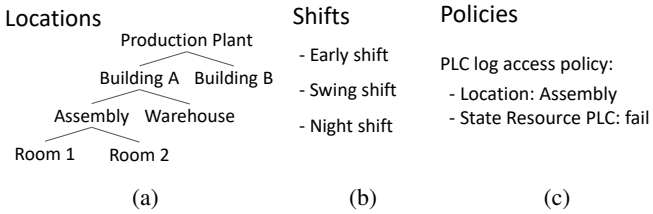

Fig. 3: Location hierarchy, shifts, and policies example.

model the current state of an entity in the *SubjectModel*, like in our example in Fig. 1. The other is to define access control rules (policies), defining the minimum required properties for data access, to ensure confidentiality. An example policy is shown on the right of Fig. 3c. It states that to access the log files of the resource PLC, one has to be at least in the assembly area and PLC needs to be in a fail state.

For our analysis, we use a non-invasive approach of extending the DDSA analysis process described in Sec. II, with the new information provided by the Context Extension meta-model. Confidentiality analysis relevant information provided by the Context Extension meta-model is added to the same modeling entities, which are extended by the Data-Driven Extension.

V. An Approach to Context-Based Confidentiality Analysis

As our second contribution we implemented a model to model transformation, that transforms an instance of our extension to an instance of the Data-Driven Extension. This

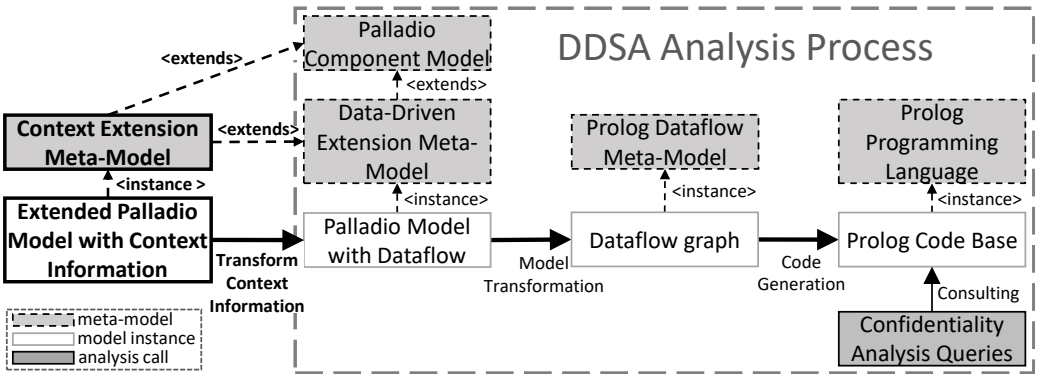

Fig. 4: Model view of the extended DDSA analysis process.

way we add our context-based confidentiality information to the DDSA analysis process while further maintaining the non-invasive approach. As shown in Fig. 4, the whole DDSA analysis process is made up of an transformation chain, transforming PCM instances with dataflow definition to Prolog code. The analysis process relies on *Characteristics* and *CharacteristicTypes*. *Characteristics* describe the properties of data or resources. These *Characteristics* hold multiple values, while the *CharacteristicType* defines the corresponding type. In its current form the analysis process can only consider fixed values for *Characteristics*, in the form of literals of an predefined enumeration. The class diagram in Fig. 5 shows the composition of the *Characteristic* specification. The running

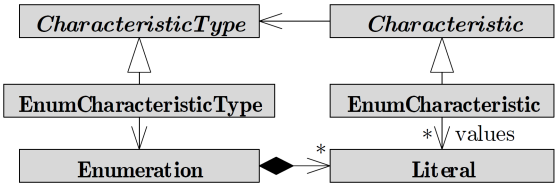

Fig. 5: Meta-model excerpt of characteristic specification [3].

example, described in [3], defines CharacteristicTypes named *AccessRights* and *Roles* that refer to an enumeration. This enumeration holds a literal for every available role, which represents the values of the corresponding *Characteristic*. *Roles* are then set as a property for *UsageScenarios* in the PCM UsageModel, while *AccessRights* are set for operations on data. As *Roles* and *AccessRights* base on the same enumeration, the values can be compared, and a role-based access control strategy is implemented.

As subject states and policies both consist of a set of contexts, they will be considered as *context sets*. These context sets build the base for a transformation. The transformation of our approach maps the context set of subjects or policies to literals of an enumeration. Subject states and policies later represent CharacteristicTypes, similar to *Roles* and *AccessRights* in [3], that base on the same enumeration.

As some contexts have dependencies that can not be represented in an enumeration literal, the dependencies have first to be resolved. To do so, the contained contexts of each subject's and policy's context set is iterated.

For a context with a hierarchical dependency, all parent nodes of the hierarchical tree, are inferred by the context, as they define a more general property. To resolve the dependency, all more general subject states resulting from the dependency have to be represented by a context set, which will have a reference to the subject. A copy of the current context set is created for each inferred context. For each copy, the original context with hierarchical dependency is swapped for one of the inferred contexts.

Considering the contexts of 'Technician' from the running example in Fig. 1 and the location hierarchy and shifts shown in Fig. 3a and 3b, resolving the dependencies results in nine context sets. As the 'Technician' has a context set that contains the location context for 'Room 1' and a shift context for 'Swing shift', the context set has to be copied and adjusted tree times. Once for each parent node when iterating the hierarchical tree to its root, which in our example is 'Assembly', 'Building A', and 'Production Area'. The copies contain their respective inferred location context, as well as the shift context for 'Swing shift'.

Environment contexts are used to define a dependency to a property of an entity in the system. They are mainly used for creating policies. By resolving their dependency, they are always removed from the context set, as they otherwise do not provide any additional value for the further DDSA process and analysis. The dependency of an environment context is resolved by checking whether the referenced entity satisfies the target attributes defined in the context. As the referenced entities are fully modeled at the time of resolving dependencies and will not change during the DDSA process, the entities attribute is compared to the target attribute. If the defined target attribute is satisfied, the context is removed from the context set. If the attributes can not be satisfied, changes to the current context set need to be made, depending on the type of environment context. When used in a policy, unsatisfied StateContexts remove the whole context set they are contained in from the transformation, because the whole policy could never be satisfied. While the policy still exists in the model, it is thereby disregarded for the further DDSA analysis process. For a subject in the system, this might result in an access

restriction.
In our example, the dependency of 'PLC log access policy' from Fig. 3c needs to be resolved. As 'PLC' is in a fail state, as shown in 1, the StateContext from the policy can be removed from further processing.
A pseudocode representation of the process is shown in line

Algorithm 1 Resolve context dependencies and create enumeration to represent properties.

```
1:  contextSets = ∅
2:  for all Subjects do
3:      inferredContextSets ← subject.contexts
4:      while inferredContextSets ≠ ∅ do
5:          for each inferredContextSet in inferredContextSets do
6:              inferredContextSets.remove(inferredContextSet)
7:              for each context in inferredContextSet do
8:                  if context is hierarchical then
9:                      inferredContextSets.add(hierarchical inferred sets)
10:             contextSets += inferredContextSet
11:             contextMappingTable.insert(subject, inferredContextSet)
12: for all Policies do
13:     for each context in policy.contexts do
14:         if context is environmental then
15:             if environmental dependency is given then
16:                 policy.contexts.remove(context)
17:             else
18:                 skip processing current policy
19:     policySet = contextSets.getMatchingSet(policy.contexts)
20:     if policy.contexts do not match a set in contextSets then
21:         policySet = new ContextSet(policy.contexts)
22:         contextSets.add(policySet)
23:     policyMappingTable.insert(policy, policySet)
24: for each contextSet in contextSets do
25:     literalMappingTable.insert(contextSet, new EnumLiteral)
```

1 - 23 in Algorithm 1. While resolving the hierarchical dependencies in line 5 - 11, a new context set is additionally created for each possible subset of contexts in each context set. When creating a new contexts set, already existing context sets are iterated and matched, to prevent the creation of multiple context sets containing the same contexts. This way, each occurring context combination is represented by a single unique context set without dependencies. The resulting context sets for each subject are saved using a mapping table. After the dependencies of a policy are resolved in line 13 - 18, the policy's context set is matched to the unique context sets. If no matching can be found, a new unique context set is created specifically for the policy. A mapping table is used to save the resulting context sets of each policy.
In lines 24 to 25 of Algorithm 1, the mapped context sets of subjects and policies are mapped to a enumeration literal, forming an enumeration that can be used for the DDSA analysis process. By replacing the context sets with the corresponding literals, the modified model instances can than be analyzed by the DDSA analysis process, without making any changes to the analysis process itself. For our example, the contexts of 'Technician', 'Worker', and 'PLC' after the transformation are shown in Fig. 6. For easier readability, the literals are named according to the contexts in the context set they represent.

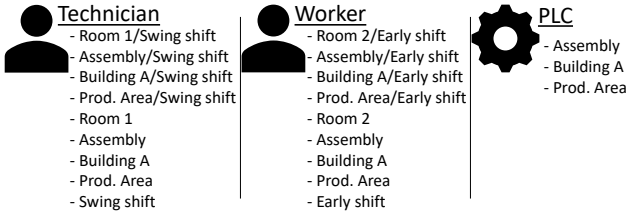

Fig. 6: Subjects of running example after transformation.

VI. Evaluation

The objective of this evaluation is to verify, that the general functionality of our approach is working as intended. To do so, the accuracy of the proposed model transformation and the scalability of the whole analysis process are evaluated. Our assumptions concerning improved usability and reduced error-proneness are not evaluated in this paper. To gain a well-designed evaluation structure, we used the goal question metric approach [22]. The evaluation, thereby, can be split up in two evaluation goals:

- *EG-1:* Examine the transformation accuracy in converting modeled information.
- *EG-2:* Examine the scalability of the analysis process when analyzing large systems.

A. Design of Evaluation of Transformation Accuracy

As we want to assure that the extended DDSA analysis process still works accurately, the accuracy of our newly added transformation needs to be evaluated. The resulting evaluation question *Q-1* is whether the transformation can accurately transform context information, to be used in the DDSA analysis process. For evaluating the accuracy of our transformation, we compared the resulting DDSA model after executing the transformation to the expected model. We implemented 17 minimalist scenarios, which cover the possible context combinations and equivalence classes of context dependencies. After using our transformation on the models, we compare the results with manually created models that represent the same properties. As a base, we use the same PCM model of the *distance tracker* case study, that has been created and extended to evaluate the DDSA approach [3]. The *distance tracker* case study consists of an app that tracks the user, aggregates coordinates, and sends this information to a tracking service, which shall not receive raw coordinates but only the distance. This distance tracker application could be a part of the industrial use case described by Al-Ali et al. [23], where, in certain cases, the supervisor of a shift can see the distance of workers to their workplace. We extend the model with our approach and add various context information. All contexts that are created for a scenario are used to define one policy, as well as to represent a single user's current state. The user is set to perform a minimalist usage behavior, with six calls to the system, as defined in the usage model. The policy

is applied to the repository model component resembling the distance tracker service.

To evaluate the transformation of hierarchy contexts, we create a binary context hierarchy tree. We refer to the root node as '*General*'. The root is connected to two nodes. One of them we call '*Middle*'. '*Middle*' is connected to two further nodes. We refer to one of them as '*Specific*'. Using this hierarchy, we create three scenarios. For scenario *S1* the context set is only made up of the context for '*General*'. The context set for *S2* contains the context for '*Middle*' and *S3* for '*Specific*'.

To evaluate the transformation of environmental contexts, we define two scenarios. *S4* contains an environmental context that defines a target attribute that is not satisfied, while the target attribute in *S5* is satisfied. We create an enumeration with two literals and represent one of the literals with a simple context for scenario *S6*. The remaining scenarios are all possible combinations of the previous scenarios. *S7* to *S12* combine the scenarios for hierarchical contexts and environmental contexts. While *S13* to *S17* combine *S1* to *S5* with the scenario for simple contexts *S6*.

As our contribution is strongly based on the DDSA approach [3], we use the same metrics that have been used to evaluate the accuracy of the DDSA approaches analysis, to evaluate our questions concerned with *EG-1*:

The precision (TPF) metric *M-1.1* calculates the ratio of correctly created and set enumeration literals t_p and number of enumeration literals P that are set in the reference models - $TPF = \frac{t_p}{P}$. [24]

The recall (PPV) metric *M-1.2* calculates the ratio of correctly created and set enumeration literals t_p and the sum $t_p + f_p$ of correctly created and falsely created enumeration literals - $PPV = \frac{t_p}{t_p + f_p}$. [24]

B. Design of Evaluation of Analysis Scalability

Transforming a system model which has been extended by our approach to the data-driven model, can result in an enumeration with a high amount of literals. This is especially true when a lot of users, contexts, and policies have been modeled. Especially if looking at organizations with multiple business associates and IIoT interconnectedness, the models can increase in size and complexity. To achieve *EG-2*, we want to examine how individual parts of the extended analysis process scale with big enumerations used for access control. The size of the enumeration created by our transformation depends on the context sets that result from the information added by our context-based extension. The amount of context sets created by our transformation depends on two dimensions. The number of modeled users and policies, as well as the depth of hierarchies of subjects or contexts.

Our first two questions are how the model transformation times scale when increasing the number of subjects (*Q-2.1*) and policies (*Q-2.2*). While question *Q-2.3* covers how the model transformation times scale when increasing the depth of hierarchic properties. The time includes the steps for reprocessing such as solving the environmental contexts.

As a foundation to answer the questions, we use the same *distance tracker* case study PCM model, that has been used for evaluating goal *EG-1*. For question *Q-2.1*, we create a Context-Based Extension model with an increasing number of users $(10^0, 10^1, 10^2, 10^3, 10^4)$. Each user has a single *simple context* as a property that is unique to them. All users are set to perform a minimalist usage behavior, with six calls to the system, defined in the usage model. For question *Q-2.2*, we increase the number of policies. Similar to the users created for *Q-2.1*, each rule is made up of a single unique *simple context*. All policies are applied to the repository model component resembling the distance tracker service. The setup for question *Q-2.3* is mainly made up of a hierarchical property tree, with increasing depth. The nodes of the hierarchy tree only have one child node each. We create a single *hierarchical context* resembling the bottom leaf of the hierarchical property tree and a user that has that context. We set the user to perform the same minimalist usage behavior used in the setup of *Q-2.1*.

We use the time needed by each transformation run, as metric *M-2.1*, to evaluate the questions concerned with *EG-2*. The run times are measured for each transformation and increment of scaling individually, excluding load times of the models and are plotted to have a better view of the general tendency of the runtime behavior. We average *M-2.1* of multiple runs with the same workload to get a mean value for a certain workload, with good variance. We chose this way of evaluating the scalability, as it is established and has been used to evaluate other projects or approaches realized with the PCM, like for example the iObserve approach [19].

C. Results of Evaluation of Transformation Accuracy

The results for the transformation accuracy show, that for all scenarios the transformation was able to maintain a TPF and PPV value of 1.0. A TPF value of 1.0 suggests, that the transformation has created and set correct literals in the resulting models. A PPV value of 1.0 suggests, that the transformation did not falsely create and set more or less literals than expected.

These results suggest, that the transformation, as described in section V, correctly converts the context information added by our approach. Additionally, this also suggests that the DDSA process will be able to run the resulting model, given that a correct PCM model is extended by our approach.

D. Results of Evaluation of Analysis Scalability

The result for scaling the number of users in the modeled system (Q-2.1) shows that the transformation of our context-based extension shows a superlinear runtime behavior. This is as expected, due to matching of each user's context set, to prevent the creation of multiple literals with the same meaning. We estimate a worst-case time complexity of $O(n^2)$. This means the runtime behavior might scale quadratically for higher numbers of users. The first transformation of the DDSA process shows sublinear runtime, as shown in Fig. 7.

The results of scaling the number of applied policies (Q-2.2) shows a linear runtime behavior for $< 10^3$ policies, for

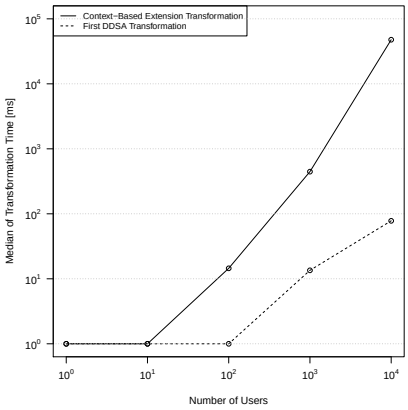

Fig. 7: Execution time with increasing amount of users.

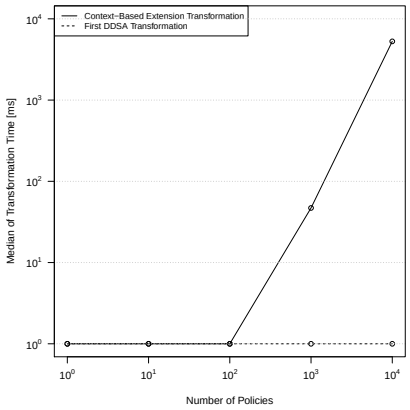

Fig. 8: Execution time with increasing amount of policies.

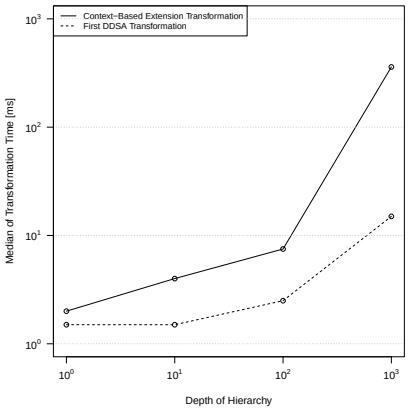

Fig. 9: Execution time with increasingly deep hierarchies.

the transformation of our context-based extension (see Fig. 8). This was to be expected, as policies are only matched to the already transformed context sets. The first DDSA transformation shows constant runtime behavior. We suspect, while each user's dataflow is transformed separately for Q-2.1, the first DDSA transformation treats a set of applied policies as a single entity for Q-2.2.

When increasing the depth of a property hierarchy (Q-2.3), the context-based extension transformation of our approach and the first DDSA transformation show a similar execution time behavior (see Fig. 9). Both have a linear behavior up to a hierarchy depth of 10^2 but show a larger increase in the needed time, suggesting an exponential runtime for deeper hierarchies. When looking at the hierarchical properties we implemented for this paper, we assume that it is unlikely for hierarchies to exceed a depth of 10^2. Especially when focusing on hierarchical trees for locations, there is a physical limit on how fine-granulated a location can be modeled.

The case of application, which the analysis process is designed for, currently only focuses on modeling and analysing predefined scenarios. Even though the results suggest, that our approach adds an overhead, that is double the runtime of the first transformation of the DDSA process, for the use case this is still acceptable.

E. Threads to Validity

As the accuracy of our approach is evaluated with a case study, we discuss the internal and external validity of our study, characterized by Runeson, Höst, Austen and Regnell [25].

Internal validity ensures that causal relations are valid, i.e. the factor that is expected to have an influence is the only influencing factor. In our study, we evaluate the accuracy and scalability of our implemented model to model transformation. We expect the creation of base models, selection of scenarios, and the model layer only evaluation to be the main influence factors. We mitigated the model creation bias by using one of the case studies defined by the iFlow approach, which has already been used to evaluate the DDSA approach, instead of creating our own. The selection of scenarios is derived from equivalence classes derived from the models in our approach. In turn, the structure and design of our models are abstracted from technical reports, as described in Sec. IV. This mitigates the selection bias, as the equivalence classes used are also based on the general descriptions in the technical reports. We, therefore, expect that the selection is well-founded. The accuracy of the resulting DDSA model was only validated on a model layer. However, we assume that since they are valid input models for the DDSA analysis, that the corresponding results will also be correct. The DDSA analysis itself was evaluated in [3]. Therefore the results should be valid. For the scalability analysis, we make the same assumption. Kunz [26] suggests that the transformation to Prolog scales at least linearly. Hence, we assume that the DDSA process can handle our increased number of enumerations.

External validity ensures that the findings can be applied to other situations, groups, or events and that the results are valuable for others. We believe that the results of our evaluation hold for cases that base on the content of the technical reports, as well as cases that allow representing its confidentiality relevant properties with the tree context types defined in Sec. IV. Additionally, we used precision, recall, and execution time for the results of the evaluation. These metrics provide no subjective interpretation by a researcher.

F. Assumptions and Limitations

The fundamental assumption of our approach is, that all possible contexts are known to a software architect that describes a system. Dynamical changes in contexts during runtime can not be regarded by our approach if they are not known during design time. However, we assume that if a set of policies is already known during design time, that in turn all necessary contexts are known as well. The properties used for describing the policies will match the minimum of contexts since our approach mainly uses contexts to form policies. Changes of contexts outside of the minimum set do consequently not carry information used for the analysis process, and are thereby

optional.

Our main limitation is that currently there is no way to represent active context changes of e.g. a subject within the usage description of the PCM. A usage scenario that includes multiple context changes, e.g. of a subject's location, needs to be split into separate sub-scenarios at each context change. The model needs to be adjusted accordingly and each sub-scenario is analyzed individually. While this limitation lowers the usability of our approach, the overall analysis result of a scenario stays the same.

VII. Conclusion

In this paper, we proposed our approach to representing context-based confidentiality properties for a dataflow analysis on the architectural level. We introduced a metamodel for defining confidentiality relevant information. The metamodel extends the Data-Driven Software Architecture (DDSA) modeling language, which is an extension of the architectural design language PCM. We extended the existing DDSA analysis process to use the confidentiality information for its dataflow analysis by adding a model to model transformation. The transformation converts the extended metamodel information back to the plain DDSA modeling language. This way, the DDSA analysis process can be used without any changes done directly to it. The evaluation examines transformation accuracy and scalability. The transformation converted all scenarios accurately and scaled as expected.

The benefit of our approach is that we assume that software architects that conduct confidentiality analyses in the design phase have a more natural and less error-prone way of defining access rights. Also, the model extension serves as a more clearly laid out documentation of access rights, that can be used for communication with other stakeholders.

In future work, we aim to evaluate our assumption concerning usability and error-proneness in creating context-dependent access control policies with a user study, as well as evaluating the applicability of our approach using an automated implementation of the whole DDSA process. We plan to improve the scalability of the transformation and expand the variety of considered contexts. In the long run, we aim to integrate the extended DDSA approach in an approach for automatic system adaptation and evolution through run-time observation and continuous quality analysis.

VIII. Acknowledgments

This work is supported by the DFG (German Research Foundation), grant number HE8596/1-1 (FluidTrust).

References

[1] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (iiot): An analysis framework," *Computers in Industry*, vol. 101, pp. 1 – 12, 2018.

[2] V. Roblek, M. Meško, and A. Krapež, "A complexity view of industry 4.0," *SAGE Open*, vol. 6, no. 2, 2016.

[3] S. Seifermann, R. Heinrich, and R. Reussner, "Data-driven software architecture for analyzing confidentiality," in *2019 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2019, pp. 1–10.

[4] R. Reussner, S. Becker, J. Happe, R. Heinrich, A. Koziolek, H. Koziolek, M. Kramer, and K. Krogmann, *Modeling and simulating software architectures: The Palladio approach*. Cambridge, Massachusetts: MIT Press, 2016.

[5] S. Becker, H. Koziolek, and R. Reussner, "Model-based performance prediction with the palladio component model," in *Proceedings of the 6th International Workshop on Software and Performance*, ser. WOSP '07. ACM, 2007, pp. 54–65.

[6] S. Seifermann, "Architectural data flow analysis," in *Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture*, ser. WICSA'16. IEEE, 2016, pp. 270–271.

[7] A. A. E. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin, "Organization based access control," in *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, 6 2003, pp. 120–131.

[8] D. Ferraiolo, J. Cugini, and D. R. Kuhn, "Role-based access control (rbac): Features and motivations," in *Proceedings of 11th annual computer security application conference*, 1995, pp. 241–48.

[9] F. Cuppens and A. Miège, "Modelling contexts in the or-bac model," in *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, no. 19, 12 2003, pp. 416–425.

[10] F. Autrel. (2018) Motorbac. [Online]. Available: http://motorbac.sourceforge.net/

[11] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attribute-based access control," *IEEE Computer*, vol. 48, no. 2, pp. 85–88, 2 2015.

[12] P. H. Nguyen, M. Kramer, J. Klein, and Y. Le Traon, "An extensive systematic review on the model-driven development of secure systems," *Information and Software Technology*, vol. 68, pp. 62–81, 2015.

[13] F. Swiderski, *Threat modeling*. Microsoft Press, 2004.

[14] J. Jürjens, "Umlsec: Extending uml for secure systems development," in *International Conference on The Unified Modeling Language*. Springer, 2002, pp. 412–425.

[15] A. S. Ahmadian, D. Strüber, V. Riediger, and J. Jürjens, "Model-based privacy analysis in industrial ecosystems," in *European Conference on Modelling Foundations and Applications*. Springer, 2017, pp. 215–231.

[16] T. Lodderstedt, D. Basin, and J. Doser, "Secureuml: A uml-based modeling language for model-driven security," in *International Conference on the Unified Modeling Language*. Springer, 2002, pp. 426–441.

[17] K. Katkalov, K. Stenzel, M. Borek, and W. Reif, "Model-driven development of information flow-secure systems with iflow," in *2013 International Conference on Social Computing*. IEEE, 2013, pp. 51–56.

[18] K. Tuma, R. Scandariato, and M. Balliu, "Flaws in flows: Unveiling design flaws via information flow analysis," in *2019 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2019, pp. 191–200.

[19] R. Heinrich, "Architectural run-time models for performance and privacy analysis in dynamic cloud applications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 4, pp. 13–22, 2016.

[20] R. Al-Ali, T. Bures, B.-O. Hartmann, J. Havlik, R. Heinrich, P. Hnetynka, A. Juan-Verdejo, P. Parizek, S. Seifermann, and M. Walter, "Use cases in dataflow-based privacy and trust modeling and analysis in industry 4.0 systems," Karlsruher Institut für Technologie (KIT), Tech. Rep. 9, 2018.

[21] S. Seifermann and M. Walter, "Evolving a use case for industry 4.0 environments towards integration of physical access control," in *Proceedings of the Workshops of the Software Engineering Conference 2019, Stuttgart, Germany, February 19, 2019.*, ser. Softwaretechnik Trends, 2019, pp. 106–108.

[22] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.

[23] R. Al-Ali, P. Hnetynka, J. Havlik, V. Krivka, R. Heinrich, S. Seifermann, M. Walter, and A. Juan-Verdejo, "Dynamic security rules for legacy systems," in *Proceedings of the 13th European Conference on Software Architecture - Volume 2*. ACM, 2019, p. 277–284.

[24] D. M. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[25] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.

[26] J. Kunz, "Efficient data flow constraint analysis," Master's thesis, Karlsruhe Institute of Technology (KIT), Germany, 2018.