Modeling of Dynamic Trust Contracts for Industry 4.0 Systems[*]

Rima Al-Ali
Charles University
Prague, Czech Republic
alali@d3s.mff.cuni.cz

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

Petr Hnetynka
Charles University
Prague, Czech Republic
hnetynka@d3s.mff.cuni.cz

Adrian Juan-Verdejo
CAS Software
Karlsruhe, Germany
adrian.juan@cas.de

Stephan Seifermann
Karlsruhe Institute of Technology
Karlsruhe, Germany
stephan.seifermann@kit.edu

Maximilian Walter
Karlsruhe Institute of Technology
Karlsruhe, Germany
maximilian.walter@kit.edu

ABSTRACT

Due to their close relation to physical and virtual entities (humans, machines, processes, etc.) including their changing state and context, modern cyber-physical and IoT systems exhibit a high degree of architectural dynamicity. While sharing of data among all the entities of the system is the key driver to the efficiency of the system, it is at the same time necessary to effectively control which data are shared, with whom, and in which context so as to prevent potential misuse. The problem however is that traditional methods to security and privacy, which typically rely on rigid hierarchies, cannot easily cope with the high degree of architectural dynamicity. In this paper, we outline an approach to ensure security and privacy on the architectural level in systems with dynamic architectures. In particular, we focus on a) data tracking using data flows and data processing described in system architectures, b) descriptions of dynamic sharing scenarios including decision derivation based on the current situation, and c) a runtime analysis platform that regulates data exchange. We ground the approach and illustrate it in the Industry 4.0 setting, as this is the domain in which we apply our approach as part of our project Trust 4.0, but we believe it can be used in other applications domains as well.

CCS CONCEPTS

• **Applied computing → Supply chain management**; • **Security and privacy** → *Domain-specific security and privacy architectures*;

KEYWORDS

Industry 4.0, confidentiality, security enforcement

[*]This work was supported by the German Federal Ministry of Education and Research (grant numbers 01IS17106A and 01IS17106B), the Technological Agency of the Czech Republic (project no. 2017TF04000064) and by Charles University institutional funding SVV.

ACM Reference Format:
Rima Al-Ali, Robert Heinrich, Petr Hnetynka, Adrian Juan-Verdejo, Stephan Seifermann, and Maximilian Walter. 2018. Modeling of Dynamic Trust Contracts for Industry 4.0 Systems. In *12th European Conference on Software Architecture: Companion Proceedings (ECSA '18), September 24–28, 2018, Madrid, Spain.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3241403.3241450

1 INTRODUCTION

Modern cyber-physical and IoT [8] systems (CPS) [5] are distributed software-intensive systems that combine mobile devices with server-side computation. The applications of CPS are numerous ranging from smart homes, smart cities, smart electric grids, smart mobility to the recent initiative of optimizing and customizing production and industrial processes, termed Industry 4.0 [10], which enacts ad-hoc cooperation between machines, humans, and organizations in supply and production chains.

An important aspect of a CPS is that its entities are bound to physical entities such as humans (via smartphones or other end-user terminals), machines, and virtual entities such as suppliers, producers, and processes. This implies that the architecture of a modern CPS becomes a rather complex structure, which dynamically changes (i.e. entities and communication emerge and disappear) based on movement, ad-hoc cooperation and interactions, as well as constantly changing context of entities. This is further exacerbated by changing goals and behavior of the entities, which for instance in Industry 4.0 comes from the very high variability of the production process.

An important problem in this settings is that the inherent architectural dynamicity contradicts with current privacy and trust mechanisms and makes them unsuitable since they are built around assumptions of rigid hierarchical architectures. To overcome this issue, overall behavior of a system has to be described and bound to a set of situations reflecting the dynamic cooperations and context. This behavior then needs to be reflected in privacy and trust mechanisms.

In this paper, we propose a novel approach to privacy and trust tied to the dynamic architecture of the ad-hoc dynamic cooperation. We establish privacy by ensuring confidentiality according to policies. This establishes trust between data sharing parties because the system prohibits information leaks. Though it applies to the whole class of modern CPS, we ground it in the Industry 4.0 as we develop it in the frame of our project Trust 4.0, which focuses on trust in Industry 4.0.

Our approach combines the architectural modeling of a system in the Palladio Component Model (PCM) and the concept of autonomic entity ensembles to capture dynamic situations in the system and related privacy and trust requirements. In the paper, we show how to model a system with our approach and additionally, we discuss our preliminary work on runtime analysis and enforcement of the security and privacy in the dynamic context along with our initial prototypes and emerging results. The paper concludes with a research road-map.

2 RUNNING EXAMPLE

As a running example for the paper, we use a case stemming from real deployments of ValueStreamer[7]; that is, a commercial product by CAS software company – co-authors of this paper. ValueStreamer is a shop floor management system that delivers lean management principles and data sharing across the entire supply chain (SC). In the running example, we consider ValueStreamer to be used by a car manufacturer and its entire SC including multiple potential suppliers of car brakes and tires. The ValueStreamer delivers data — such as error rates of production machines — to stakeholders within and across organizations. The knowledge of error rates allows stakeholders to monitor, predict, and optimize the production process. However, the error rates constitute sensitive information; hence, the ValueStreamer has to control access to them.

The permission to share data depends on the context as well as on the particular business process involved and its stakeholders' roles, time and date, the participant SC entity and its members, the specific time window for data sharing, the sharing location, and the provenance of the data across SC. For instance with respect to production machine error rates: (1) different departments within the brake supplier have heads of department with access to error rates of machines of his/her particular department but not to error rates of machines in other departments of the same factory. (2) The head of engineering responsible for a particular manufacturing process has access to error rates but only of machines involved in the process and time-wise related to the process. (3) A subcontractor's technician fixing a machine may see its detailed rates log but only together with a technician of the department responsible for the machine. (4) The brake supplier wants to share its error rates with the head of a subsidiary company — to allow for quality improvement — but only after proper anonymization to not reveal details of the manufacturing process.

Note that while scenario #1 is a rather traditional case where a static organizational structure determines data sharing, scenario #2 and #3 are rather dynamic because they depend on the dynamically changing manufacturing process. Additionally, scenario #3 allows sharing only within the context of a coordinated action involving multiple stakeholders. Scenario #4 adds in the topic of data aggregation and anonymization,like only sharing aggregated values over a certain time frame, whereby organizations can only share data with a certain privacy level (i.e. they are not too sensitive).

3 STATE OF THE ART

There is a lot of research in the area of testing and establishing privacy of software systems. Our approach, especially, aims for preserving confidentiality of specific data in software design, and software operation. Therefore, the areas of confidentiality analyses, access control, security enforcement and data sensitivity are closely related to our work.

We distinguish confidentiality analyses by the development phase they take place in. During design time, there are high-level approaches such as Threat Modeling [20] that check coarse-grained system descriptions for security degrading flaws in a creative process. In case of changes, this is cumbersome because they do not use the same models as architects and developers. Hoisl et. al. [11] use object flows to ensure confidentiality and integrity during object transmissions in service-oriented architecture designs. They, however, do not analyze access rights but only ensure transmission of sensitive data via secured links. UMLSec [12] ensures secure communication as well but also considers role based access control. The definition of confidentiality takes place on the control rather than the data flow, which complicates formulating access control constraints.

During the implementation, information flow analyses such as Joana [19] or code verification approaches such as KeY [2] detect information leaks. Both, however, require the actual implementation to be available as source code. Additionally, the approaches do not scale well for large, complex industry 4.0 scenarios but require partitioning the analyzed system. They are also specific to one programming language. In heterogeneous environments like they exist in an industry 4.0 setting, they are not applicable.

Approaches such as SecureUML [13] provide access control specifications. A modeler specifies roles, permissions, and users in a stereotyped UML diagram. The specification happens at design-time, which is problematic in Industry 4.0 environment, where roles can change rapidly and multiple different parties interact.

In security enforcement, we see relations to approaches building new enforcement platforms, and approaches for generating inputs for existing platforms. MDSE@R [3] is a combination of security specification and enforcement platform. The enforcement platform uses aspect-oriented programming to include security in existing applications without changing their source code. Even if new platforms are promising, we aim for integrating our approach into existing enforcement platforms and generate inputs for them.

Another research field in industry 4.0 is data sensitivity. This was considered in the QoCIM framework [15], which is based on Quality-of-Context (QoC) analyses considering the impact on sharing different levels of sensitive data. The researchers define trust-based context contracts [14], which are dynamic and developed independently of the contract sides. Furthermore, the contract changes could be in modification of trust, visibility circles, and context data. They, however, define trust in terms of quality of meta-data rather then in terms of confidentiality.

A common problem of all the approaches above is that they lack the ability to capture highly dynamic situations where multiple parties jointly coordinate together towards a common goal. We further combine this with the notion of privacy levels to enable sharing of data that has been properly anonymized.

4 SOLUTION OVERVIEW

Our approach features two types of models: (1) models of data flows and data manipulation, as well as (2) models of privacy requirements bound to dynamic situations. We constantly update these models during runtime to reflect dynamic changes in the system and its execution context. Models of data flows (1) serve for describing the system and the processes in it. This model is inherently dynamic in the sense that it describes the individual data flows and that the overall system behavior emerges as a result of instantiation of the data flows depending on the current situation (i.e. the context which consists of current suppliers, products currently manufactured, etc.). Models of privacy requirements (2) use the same notion of a situation and connect it to statements about permitted and disallowed sharing of data. Having these two types of models, we perform a runtime analysis (which itself can be considered the third constituent of our approach), in which we track how data flows dynamically emerge and ensure that the data flows comply to the contextual privacy requirements. In the subsection below, we briefly exemplify these three parts of our approach.

4.1 Models of Data Flows

We describe and analyze the systems on an architectural level using the established Palladio Component Model (PCM) [17]. The PCM is a collection of multiple models that describe the system, its behavior, and its usage. A repository model describes software components (i.e. reusable building blocks of software), their required and provided interfaces, data types, and the component behavior. A system model describes offered services and how the system provides them by connecting instances of components. A resource model and an allocation model describe available hardware and deployment of components to this hardware. Usage models describe how users use the system by calling offered services. A data flow modeling extension [18] allows specifying data processing in more detail. PCM subsystems allow representing multiple connected systems and their data interchange.

We use this behavior description to determine security properties of data processed in systems that confidentiality contracts can use in their specification. The following paragraphs describe the behavior model, while Subsection 4.2 describes the contracts. We cover the actual analysis in Subsection 4.3. In our explanation, we use scenario #4 and its simplified illustration in Figure 1.

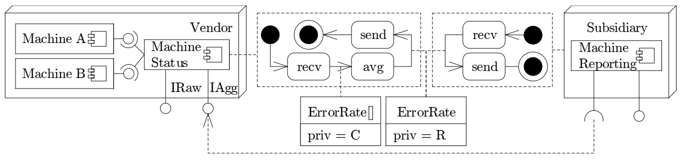

Figure 1: Simplified system architecture including data processing.

The basic idea of data flow modeling in PCM is making data instances explicit and extending existing actions in control flows with explicit data acquisition, emission, and processing. Data instances have a certain data type and additional characteristics that, for instance, can describe access rights, granularity of sensor data, owners, or a privacy level. In Figure 1, we use the data type *ErrorRate* and the privacy characteristic *priv*. Possible privacy levels are confidential *C* and restricted *R* in this example. Data acquisition and emission can take place on service parameters, return values, or internal component stores. Data processing consists of a chain of data processors that take input data and produce output data. In our example, the error rates are gathered from the machines. Afterwards, the *MachineStatus* component calculates the average of the error rates and sends this data to the calling *MachineReporting* component. Each data processor can have an effect on the characteristics associated with the data instance. In our example, the *avg* operator takes a list of error rates and converts it into an average error rate per machine. This reduces the confidentiality level from confidential to restricted. In this way, we derive the confidentiality level of data at a given location in the architecture from the modeled behavior.

4.2 Models of Trust Contracts

To reflect the architectural dynamics of the Industry 4.0 processes in the definition of privacy and trust contracts, we exploit the concept of autonomic entity ensembles [4]. An ensemble is an architectural construct that models a group of entities, which dynamically coordinate to achieve a common goal. An entity can be a PCM software component but can be a domain concept such as a person using the system as well.

A typical example of an ensemble in the smart cyber-physical domain would be a group of cars that jointly coordinate to discover and reserve parking places. The definition of an ensemble typically consists of two parts: (a) *membership constraint*, which expresses which entity instances and under which condition form ensembles, and (b) *group goal*, which captures the functional purpose of the ensembles. Typically, the group goal would be to exchange data or to perform a coordinated action. The same ensemble definition can be instantiated simultaneously multiple times, which naturally corresponds to the fact that different groups of entities at different places and times can pursue a similar goal.

In the context of trust in Industry 4.0, we use an ensemble to represent trust contracts between particular domain entities (e.g. persons, organizations, places, equipment) in a given situation. The membership constraint describes which particular entities (i.e. instances) and in which particular situation they participate in the ensemble. The group goal in this case is the enforcement of privacy. This takes the form of permission of certain data sharing between the entities participating in the ensemble.

This is illustrated in Listing 1 and 2, which show scenarios #3 and #4 respectively. Due to space constraints, we omit the declaration of entities and the ensembles for scenarios #1 and #2. They can be found at [1]. The membership is represented by the `role` selector, which identifies entities that can act in the ensemble based on their types and properties. Additional constraints over the entities can be specified in the optional `where` section. The permitted sharing is defined by `allow`, which defines the subject, object, data, and an optional required privacy level.

4.3 Trust Analysis

The goal of the trust analysis is to provide decision making for trust enforcement platforms. The analysis consists of two steps: First,

```
class SharingWithServiceman(val machine: Machine) extends Ensemble {
  val servicemen = role(entities.select[Person].filter(_.hasRole{ case Serviceman(machine) =>
          true } ))
  val accomp = role(entities.select[Person].filter(_.hasRole{ case
          Technician(machine.department) => true } ))
  where( servicemen.all(svc => accomp.some(acc => svc.location == acc.location)) )
  allow(servicemen, machine, "errorRates")
}
```

Listing 1: Ensemble-based specification of sharing rule in scenario #3

```
class SharingWithSubsidiary(val company: Company) extends Ensemble {
  val machines = role(entities.select[Machine].filter(_.company == company))
  val persons = role(entities.select[Person].filter(
    _.hasRole{ case HeadOfCompany(subsidiary) => subsidiary.parentCompany == company } ))
  allow(persons, machines, "sum(errorRates)", PrivacyLevel.RESTRICTED)
}
```

Listing 2: Ensemble-based specification of sharing rule in scenario #4

a PCM runtime analysis determines system characteristics such as privacy levels of emitted data. Second, the Trait-based Coalition Formation Framework (TCOOF) [6] evaluates ensembles and derives decisions by means of access permissions. Given the work-in-progress status of our approach, we did not evaluate the fully integrated trust analysis yet but describe the envisioned analysis.

The PCM runtime analysis determines the characteristics of a given data at a given location in the architecture. In scenario #4, the characteristic is the privacy level. The location in the architecture is the transmission link between the systems of a vendor and a subsidiary. The analysis determines the characteristics by collecting the data processors of all possible paths this data can have taken to the location and calculating the effect of the processing to the characteristics. For instance, applying an average operator can reduce the privacy level of data. To cope with dynamic changes of the systems and their execution contexts, we use an approach to update the initial models during runtime [9] based on observed changes. Thereby, we can consider changes in deployment, assembling, and data processing in the involved systems. All of these changes can have an effect on the data characteristics and are important for consistent decision making.

The actual decision making for trust contracts essentially consists of two steps: (i) determining which contracts are relevant in the current context, and (ii) deciding on allowing/denying data sharing based on the relevant contracts. (i) is a relatively novel problem specific to systems with a high degree of dynamism (as is the case of Industry 4.0). In our approach, we address (i) by transforming the ensemble definitions to a constraint solving problem. In this problem, the membership constraints form constraints of the problem. The available entities (i.e. instances) form variable domains and membership of an entity in a specific ensemble is reflected as a variable in the constraint problem. For initial results here, we exploit our existing framework TCOOF, which we have adapted for supporting the security-oriented DSL as shown in the examples above. The code can be found at [1]. It allows us to capture the ensembles via internal Scala DSL (as already shown in Listing 1 and 2). The framework evaluates the ensembles using available context information such as the privacy level of data to be shared and provides the answer to which trust contracts are currently active and to which entities they apply. Internally, TCOOF employs the Choco constraint solver [16]. The use of the solver allows for keeping the complexity of evaluation manageable and benefit from state-space pruning algorithms employed in the solver. The actual decision (ii) is derived from the results of the active ensembles.

5 CONCLUSIONS

In this paper, we outlined the Trust 4.0 approach for trust in dynamic supply chain environments. We showed how the dynamics of domains as Industry 4.0 can be addressed in architectural modeling and reflected in security and privacy that goes beyond traditional static hierarchies of roles and support situation-dependent security that regulates coordination among multiple parties.

Our approach is still in a prototypical stage and there are already many open issues, which we need to address. Looking at the issues as a research roadmap, we see the following general problems: (1) In the future we plan to investigate how to design a domain specific language for specifying systems and security requirements, that users without software development background understand them. (2) Also we want to investigate, what security and privacy guarantees we can give at design time when the whole system is dynamic. (3) At last the we want to further inverstigate the scalibily and decentralize security and privacy.

REFERENCES

[1] Trust 4.0. [n. d.]. Ensemble-based architectural framework for sharing in collaboration groups. https://github.com/Trust40-Project/icsa2018-ensembles. accessed 10.06.2018.

[2] W. Ahrendt et al. 2016. Deductive Software Verification – The KeY Book. Springer International Publishing.

[3] M. Almorsy et al. 2012. MDSE@R: Model-driven Security Engineering at Runtime. In *Proceedings of the 4th International Conference on Cyberspace Safety and Security*. Springer, 279–295.

[4] T Bures et al. 2016. Software Abstractions for Component Interaction in the Internet of Things. *Computer* 49, 12 (2016), 50–59.

[5] Tomas Bures et al. 2017. Software Engineering for Smart Cyber-Physical Systems: Challenges and Promising Solutions. *SIGSOFT Softw. Eng. Notes* 42, 2 (June 2017), 19–24.

[6] T Bures et al. 2017. *Trait-based Language for Smart Cyber-Physical Systems*. Technical Report D3S-TR-2017-01. Dep. of Distributed and Dependable Systems, Charles University, Prague.

[7] CAS Software AG and Staufen AG. [n. d.]. ValueStreamer. http://www.valuestreamer.de/en/home/. Accessed: 15.03.2018.

[8] J. Gubbi et al. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* 29, 7 (2013), 1645 – 1660.

[9] R. Heinrich. 2016. Architectural Run-time Models for Performance and Privacy Analysis in Dynamic Cloud Applications. *ACM SIGMETRICS* 43, 4 (2016), 13–22.

[10] M. Hermann et al. 2016. Design principles for industrie 4.0 scenarios. In *HICSS'16*. IEEE, 3928–3937.

[11] B. Hoisl et al. 2014. Modeling and enforcing secure object flows in process-driven SOAs: an integrated model-driven approach. *Software & Systems Modeling* 13, 2 (01 May 2014), 513–548.

[12] Jan Jürjens. 2002. UMLsec: Extending UML for Secure Systems Development. In *UML'02 — The Unified Modeling Language*. Springer, 412–425.

[13] T. Lodderstedt et al. 2002. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *UML'02 — The Unified Modeling Language*. Springer, 426–441.

[14] S. Machara et al. 2013. Trust-Based Context Contract Models for the Internet of Things. In *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*. 557–562.

[15] P. Marie et al. 2014. The QoCIM Framework: Concepts and Tools for Quality of Context Management. In *Context in Computing: A Cross-Disciplinary Approach for Modeling the Real World*. Springer, New York, 155–172.

[16] Ch. Prudhomme. [n. d.]. Choco Solver. http://www.choco-solver.org/. accessed 10.06.2018.

[17] R. Reussner et al. 2016. *Modeling and simulating software architectures: the Palladio approach*. MIT Press.

[18] S. Seifermann. 2016. Architectural Data Flow Analysis. In *13th Working IEEE/IFIP Conference on Software Architecture (WICSA16)*. IEEE, 270–271.

[19] G. Snelting et al. 2014. Checking probabilistic noninterference using JOANA. *it-Information Technology* 56, 6 (2014), 280–287.

[20] F. Swiderski and W. Snyder. 2004. *Threat Modeling*. Microsoft Press.