Traceability of Information Flow Requirements in Cyber-Physical Systems Engineering

Christopher Gerking

Paderborn University, Heinz Nixdorf Institute
Software Engineering Research Group
Paderborn, Germany
christopher.gerking@upb.de
WWW home page:
https://www.hni.uni-paderborn.de/en/software-engineering

Abstract. The secure information flow between a cyber-physical system and its environment has evolved into a critical factor. To comply with security regulations, engineers of cyber-physical systems need to trace information flow requirements from their specification to the software design. However, due to the interdisciplinary engineering of cyber-physical systems and their inherent real-time behavior, requirements specified at a discipline-spanning level are hard to verify at the discipline-specific software design level. In this PhD project, we address this problem based on a specification of information flow requirements in model-based systems engineering. We provide a technique to verify if the real-time behavior of software design models fulfills corresponding information flow properties, and trace verification results back to the initial requirements. By establishing traceability, we enable engineers to ensure the regulatory compliance of cyber-physical systems. We intend to validate the applicability of our approach in the scope of an Industrial Internet scenario.

Keywords: requirements traceability, information flow security, systems engineering

1 Problem

Along with the 4th industrial revolution, cyber-physical production systems are expected to enable highly dynamic business relationships between companies. Business partners are no longer pre-defined, but may change on the fly depending on quality factors like cost, delivery time, or even environmental friendliness. Due to the dynamic interconnection of systems in the *Industrial Internet*, security has evolved into a critical factor because business secrets are at risk of being exposed to unknown, untrusted business partners. To avoid violations of security policies, the software that drives a cyber-physical system needs to control the information flow between the system and its environment. Thus, systems must meet strict information flow requirements.

For example, consider a production system that exchanges information with a cloud-based service market to order materials. In addition, the system has access to a confidential operating plan that is considered as a business secret. Thus, it must not be exposed to the outside world. As an example for an information flow requirement, the production system needs to avoid an illegitimate flow of confidential information from the operating plan to the public service market.

Engineers of cyber-physical systems need to consider such requirements at the design stage because mandatory regulations are expected to involve rigorous security conditions. Before a novel system may start to operate, the compliance with its regulations needs to be certified by authorities [11]. This regulatory compliance often depends on the *traceability* of requirements, which is defined as "the ability to describe and follow the life of a requirement in both a forwards and backwards direction" [7]. Traceability enables certifiers to reproduce that a system complies to its specified requirements. Thus, traceability requires engineers to trace information flow requirements from their specification to the software design.

Model-driven engineering approaches enable software engineers to verify information flow properties at the level of software design models. Nevertheless, due to several characteristics of cyber-physical systems, the traceability of information flow requirements between specification and design is hard to establish by engineers. In the following, we identify three challenges for engineers (C1-C3) to be addressed in this PhD project.

First of all, beside software engineering, multiple other disciplines are involved in the development of cyber-physical systems. An integration of these disciplines at the level of model-based *systems engineering* [14] is beneficial for providing a holistic view on a system under development. However, as shown in Fig. 1, the integration also introduces an additional, discipline-spanning conception stage, preceding the software design and implementation stages. During the system conception, systems engineers create a model that includes an initial specification of the system requirements (cf. Fig. 1). Due to the negative nature of information flow requirements (describing an illegitimate flow to avoid), a fit-for-purpose specification technique is needed.

C1: How can systems engineers specify information flow requirements at a discipline-spanning conception stage?

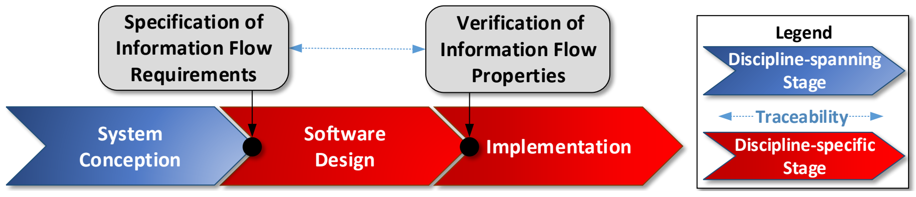

Specification of Information Flow Requirements
Verification of Information Flow Properties
System Conception
Software Design
Implementation
Legend
Discipline-spanning Stage
Traceability
Discipline-specific Stage

Fig. 1: Stages of the Engineering Process for Cyber-Physical Systems

Furthermore, due to their interaction with the physical environment, the systems need to satisfy hard real-time constraints. In order to provide sound results, verification techniques for information flow properties need to take this real-time behavior into account. Any deviation compromises the soundness of the verification and, therefore, prevents reliable requirements traceability.

C2: How can software engineers verify the information flow properties of software design models under consideration of real-time behavior?

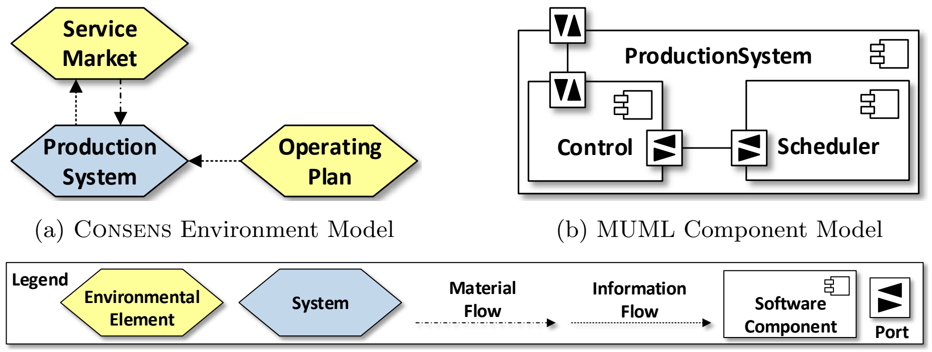
Finally, from the viewpoint of an individual discipline such as software engineering, the discipline-spanning conception leads to an increased vagueness of the requirements specification, as it relies on general terms without precise, discipline-specific semantics. Vaguely specified information flow requirements are hard to verify, even if software engineers have capable verification techniques at hand. This divergence in terms of abstraction between requirements specification and software design represents an obstacle for the desired traceability of information flow requirements as depicted in Fig. 1.

C3: How to establish traceability between the information flow requirements and the verification of information flow properties at the software design level?

2 Related Work

Today, the field of model-based systems engineering is dominated by the Systems Modeling Language (SysML, [14]). Nejati et al. [13] establish traceability between safety requirements and SysML models. In contrast, our intention to verify the information flow security (challenge C2) requires formal semantics of the underlying modeling language. Therefore, we need to consider software design models beyond the scope of SysML. Consens [3] is a method for model-based systems engineering using SysML-compliant models. Consens is capable of describing information flow between model elements, and supports the transition from systems engineering to the software design level. Therefore, Consens is a suitable basis for our needs to consider the information flow of software design models. As an example, Fig. 2a shows a Consens model of the interactions between a production system and its environment. Information is flowing from the operating plan to the system, and from the system to the service market. However, the Consens approach neither supports the specification of illegitimate information flow (challenge C1), as for example between the operating plan and the service market, nor the traceability to the software design (challenge C3).

For the transition from Consens to the software design, Gausemeier et al. [4] present a derivation of software component models based on Mechatronic-UML (MUML, [8]), a model-driven software design method for cyber-physical systems. As an example, Fig. 2b shows the software design of the production system in terms of a MUML component model. Components exchange information over ports. Whereas MUML supports the formal verification of safety



(a) Consens Environment Model (b) MUML Component Model

Fig. 2: Consens and MUML Models of the Production System

properties [5] under consideration of the real-time behavior of components, it does not enable the verification of information flow properties (challenge C2). As an example for such a property, the software design depicted in Fig. 2b needs to ensure that the Control component does not leak any confidential information about the operating plan that it receives from the Scheduler.

In general, such information flow properties are verifiable on the theoretical basis of *noninterference* [6], which states that a system is secure in terms of information flow, if the confidential inputs to a system do not affect any outputs with a lower confidentiality. If a system ensures this policy, no illegitimate flow of confidential information can ever occur. Whereas Barbuti and Tesei [2] adjust the theory of noninterference to real-time systems, their approach lacks a verification technique that is amenable to industrial-scale design models, and is ready to use by software engineers. This is in accordance with the observation by Mantel [12] that "software engineering does not respect information flow security".

A model-driven software design approach supporting the verification of information flow properties is UMLsec [10]. The authors use adversary models to describe the capabilities of attackers, and verify that the design prevents vulnerabilities that might be exploited by such attackers. Ochoa et al. [16,15] discuss explicitly the noninterference verification of UMLsec statemachine models. Houmb et al. [9] provide guidelines to ensure traceability of security requirements (challenge C3) through a systematic derivation of a UMLsec design. However, due to the missing notion of real-time, the UMLsec approach is not capable of verifying information flow properties of cyber-physical systems (challenge C2).

In summary, none of the mentioned approaches supports the specification of illegitimate information flow in model-based systems engineering, as demanded by challenge C1. With respect to challenge C2, verification techniques either disregard the real-time behavior, or lack an integration with model-driven software design. Due to these shortcomings, none of the mentioned approaches is currently suitable for tracing information flow requirements between specification and software design (challenge C3).

3 Proposed Solution

We choose MUML as the underlying software design method due to its existing support for real-time, and due to our preliminary work on the approach (cf. Sect. 4). Furthermore, we rely on CONSENS for the specification of requirements due to its tight integration with MUML. More precisely, we provide the following partial solutions to the traceability problem for information flow requirements:

Specification of illegitimate information flow at the level of CONSENS system models. To address challenge C1, we provide systems engineers with a specification technique for information flow requirements, allowing them to mark a flow between elements of a system, or its environment, as illegitimate. The specification needs to take place in a form that enables a later comparison against the actual information flow detected through verification. Thus, by distinguishing illegitimate from legitimate flow, it is possible to judge whether the information flow requirements are violated.

Deriving verifiable information flow properties from the specified requirements in an automatic fashion. As a contribution to challenge C3, the derived properties relate the initial requirements to a MUML software design model, and allow to verify the model's compliance. In order to produce meaningful verification results, the derived properties need to preserve the semantics of the initial requirements and, therefore, the derivation needs to interrelate the CONSENS system model and the MUML software design model. It is beneficial to infer the relation between these models automatically from the traces of a model transformation.

Real-time verification of noninterference properties on the basis of software design models. In order to overcome challenge C2, i.e., to decide whether a given MUML model fulfills the derived information flow properties, a rigorous verification of the noninterference needs to be carried out. To this end, we enrich the theoretical basis of real-time noninterference [2] by a ready-to-use verification technique. To cope with the infinite, real-valued statespace, we explicitly consider the applicability of existing verification techniques from the area of real-time model checking [1].

Reinterpretation of the verification results to trace them back to the initial requirements as a further contribution to challenge C3. Depending on the complexity of the interrelations between MUML software design and CONSENS system model, the verification results obtained so far are of little significance, as they do not allow the engineers to draw immediate conclusions about the initial information flow requirements. Therefore, in order to give significance to the results, we automatically relate them back to the requirements specified initially. Every specified requirement needs to be marked as met (if the non-occurrence of information flow has been proved by the verification), or as a violation (if the verification detected the occurrence of an illegitimate flow). Again, the trace of an earlier model transformation from CONSENS to MUML might contribute to the reinterpretation of the verification results by resolving the interrelations between both models.

4 Preliminary Work

Up to now, our research efforts focused mainly on the verification of MUML software design models (challenge C2) in order to provide an underlying back-end for the envisioned traceability solution. In [5], we propose the application of real-time model checking techniques in a cyber-physical system context. In order to ensure domain-specific applicability, we translate temporal logic properties from MUML to the input language of a real-time model checker, and the verification results back to MUML. However, by focusing on general temporal logic properties, the verification of information flow properties is beyond the scope of the approach up to now.

5 Expected Contributions

By overcoming challenge C2, we contribute to the verification of noninterference properties in terms of a ready-to-use, yet theorized verification technique for cyber-physical systems. In addition, we contribute a specification technique for illegitimate information flow in the context of model-based systems engineering by overcoming challenge C1. Finally, addressing challenge C3 will provide a general insight into the traceability of requirements across the boundary between discipline-spanning specification and discipline-specific verification. On the practical side, traceability enables systems engineers to receive feedback as to whether the specified requirements are met or violated by the software design. Furthermore, we also provide software engineers with a technique to verify information flow properties of design models in a cyber-physical systems context.

6 Plan for Evaluation and Validation

To evaluate our contributions, we conduct case studies on the engineering of highly interconnected cyber-physical systems with strict information flow requirements, e.g., for protecting business secrets or personal data in the Industrial Internet. Our case studies traverse the engineering process including model-based systems engineering and software design. We will define measurable quality characteristics to validate that our work meets the challenges described in Sect. 1:

C1: We contrast the requirements specification at the systems engineering level with the specification of equivalent properties at the software design level, to demonstrate the reduced effort and expertise.

C2: We compare the real-time verification against a technique without a concept of time to demonstrate the improvements with respect to the number of violations identified and false positives avoided.

C3: We provide an integrated view of systems engineering and software design to validate that our solution enables a precise distinction between violated requirements on the one hand, and requirements that are met on the other hand.

7 Current Status

We currently work towards reducing the verification problem (challenge C2) to a *refinement* check for real-time systems [8]. After preparing the verification backend, we plan to address the requirements specification at the level of model-based systems engineering (challenge C1), and the derivation of verifiable properties within the next year. Finally, we intend to carry out the final integration of systems engineering and software design in order to establish the desired traceability solution (challenge C3). Our evaluation strategy enables us to carry out a stepwise, incremental validation of our contributions.

References

1. Alur, R., Courcoubetis, C., Dill, D.L.: Model-checking in dense real-time. Information and Computation 104(1), 2–34 (1993)
2. Barbuti, R., Tesei, L.: A decidable notion of timed non-interference. Fundamenta Informaticae 54(2-3), 137–150 (2003)
3. Dorociak, R., Dumitrescu, R., Gausemeier, J., Iwanek, P.: Specification technique Consens for the description of self-optimizing systems. In: Design Methodology for Intelligent Technical Systems. Springer (2014)
4. Gausemeier, J., Schäfer, W., Greenyer, J., Kahl, S., Pook, S., Rieke, J.: Management of cross-domain model consistency during the development of advanced mechatronic systems. In: ICED 09. pp. 1–12 (2009)
5. Gerking, C., Schäfer, W., Dziwok, S., Heinzemann, C.: Domain-specific model checking for cyber-physical systems. In: MoDeVVa 2015. pp. 18–27 (2015)
6. Goguen, J.A., Meseguer, J.: Security policies and security models. In: IEEE Symposium on Security and Privacy 1982. pp. 11–20. IEEE (1982)
7. Gotel, O.C.Z., Finkelstein, A.: An analysis of the requirements traceability problem. In: ICRE '94. pp. 94–101. IEEE (1994)
8. Heinzemann, C., Brenner, C., Dziwok, S., Schäfer, W.: Automata-based refinement checking for real-time systems. Computer Science — Research and Development 30(3-4), 255–283 (2015)
9. Houmb, S.H., Islam, S., Knauss, E., Jürjens, J., Schneider, K.: Eliciting security requirements and tracing them to design. Requirements Engineering 15(1), 63–93 (2010)
10. Jürjens, J.: Secure systems development with UML. Springer (2005)
11. Kokaly, S., Salay, R., Sabetzadeh, M., Chechik, M., Maibaum, T.: Model management for regulatory compliance. In: MiSE@ICSE 2016. pp. 74–80. ACM (2016)
12. Mantel, H.: Information flow and noninterference. In: Encyclopedia of Cryptography and Security, 2nd Ed., pp. 605–607. Springer (2011)
13. Nejati, S., Sabetzadeh, M., Falessi, D., Briand, L.C., Coq, T.: A SysML-based approach to traceability management and design slicing in support of safety certification. Information & Software Technology 54(6), 569–590 (2012)
14. Object Management Group: OMG Systems Modeling Language 1.4 (2015)
15. Ochoa, M., Cuéllar, J., Pretschner, A., Hallgren, P.A.: Idea: Unwinding based model-checking and testing for non-interference on EFSMs. In: ESSoS 2015. pp. 34–42. Springer (2015)
16. Ochoa, M., Jürjens, J., Cuéllar, J.: Non-interference on UML state-charts. In: TOOLS 2012. pp. 219–235. Springer (2012)