Towards evolving secured multi-model systems with model federation

Chahrazed Boudjemila, Fabien Dagnat, Salvador Martínez
IMT Atlantique, Lab-STICC, UMR 6285
Brest, France
Email: {chahrazed.boudjemila,fabien.dagnat,salvador.martinez}@imt-atlantique.fr

***Abstract*—In order to deal with the increasing complexity of nowadays systems, model-based system engineering (MBSE) promotes the use of models all along the engineering phases. In this scenario, systems are often represented by a set of models, conforming to different modeling languages and built with different tools. This multiplicity and heterogeneity are challenging when models evolve as they may easily become inconsistent. This is even more crucial when we deal with security requirements. Indeed, being a critical property of systems, security has been integrated in MBSE so that it can be dealt with since the early phases of the project (in what is called "security-by-design"). Consequently, it needs to be taken into account as well when the system evolves in order to avoid potential security issues.**

In order to tackle this problem, we propose here to leverage on the model federation paradigm, which promotes the reification of the dependencies between heterogeneous models. Concretely, we propose the creation of a security model federation. In this federation, security related dependencies between the models representing a system are reified and equipped with security rules that can be (re)evaluated upon evolution scenarios in order to determine whether a given change impacts security. We provide an initial methodology for building such a federation and demonstrate the feasibility of the approach with a prototype implementation.

***Index Terms*—Model-based system engineering, security, model federation, model evolution**

I. Introduction

Model-based systems engineering (MBSE) is a complex system's development paradigm that promotes the use of models along all its different phases (requirements, design, analysis, verification and validation) in order to obtain improvements in productivity and quality. In MBSE approaches, systems are often composed of a collection of interdependent and heterogeneous models created by using various modeling languages and/or tools (*e.g.*, SysML, UML, BPMN, etc.). This multiplicity and heterogeneity represent a challenge when models evolve as they may easily become inconsistent. Hence, mechanisms to deal with the evolution of models are required.

Correctly managing evolution is even more crucial when we deal with security requirements. Indeed, one important area of concern within complex systems is cybersecurity. In that sense, and in order to provide to MBSE the means to deal with security since early development phases (in contrast to the common practice of adding security features later in the development process), several approaches adopt a *security by design* perspective and integrate security aspects into modeling languages (*e.g.*, UMLsec [1], SysML-Sec [2], and secBPMN [3] among others). Such security aspects need to be taken into account when models evolve in order to avoid potential security issues. While many existing approaches deal with co-evolution and consistency for at least a couple of artifacts, most of them either do not deal with security aspects [4]–[6] or do it only for a limited type of model and/or technology [1], [7]–[9]. In this sense, the focus of this paper is on the management of the evolution of multi-model systems w.r.t. their security. Two main challenges arise when managing the evolution in this scenario: 1) the semantic alignment between the security information present in different models; and 2) the discovery, management and efficient exploitation of the often implicit dependencies between models w.r.t. the security requirements of the system.

In order to tackle this problem, we propose to leverage on the model federation paradigm which promotes the reification of dependencies between heterogeneous models [10]. Reified dependencies in model federation have customizable semantics and may carry behaviours. We propose thus the creation of a *security-oriented model federation* in which the security dependencies between models are reified and equipped with security rules. When a model changes, dependencies related to the change are used to (re) evaluate these security rules and determine whether the given change impacts the security requirements of the system. The main contribution of this paper is an initial methodology for building such security-oriented model federations. We demonstrate the feasibility of our approach with a prototype implementation and its application to a case study based on the iTrust [11] system.

The remainder of this paper is structured as follows. Section II gives a brief background about *security by design* approaches and the *model federation* paradigm. In Section III, we present the details of our methodology for building security oriented model federations. We describe the application of our methodology to a case study based on the iTrust system in Section IV. Related work is discussed in Section V. Finally, we present the conclusion and outline future work in Section VI.

II. Background

We devote this section to a brief introduction to the security-by-design and model federation paradigms.

This work is partially funded by Région Bretagne.

A. *Security by design*

Security by design is a system design paradigm in which system security requirements are taken into account from the beginning of the design phase. Its main objective is to identify and address potential security vulnerabilities and risks [12] of the system under development as early as possible. In an MBSE scenario, many modeling languages and paradigms are used to describe systems. Some of the more popular ones such as UML, SysML and BMPN do not include security information by default, and thus extension mechanisms are required in order to use them in a security by design framework.

In the following we briefly describe two of such extensions, UMLsec [1], [7] and SecBPMN [3]. We focus in these two extensions as the case study we describe in Section IV rely on UML and BPMN models, but many other approaches exist. UMLSec is an extension of the UML standard. It provides a UML profile which adds security properties in the form of stereotypes (*e.g.*, *encrypted*) tagged values and application constraints. A tool using these stereotypes to analyze the security of the system is also part of the approach.

SecBPMN [3] is an extension of BPMN. Concretely, it contains the language SecBPMN-ml which contributes eight security annotations (related to security properties such as confidentiality or accountability). Annotations have a graphical syntax and have to be linked with an existing element of a BPMN model. Annotations also contain attributes.

B. *Model Federation*

The model federation approach has as its main objective the reification of a set of dependencies among a group of models [10], [13]. One notable feature of the model federation approach is that *federated models i.e.*, the models that participate in a given federation, can independently evolve in their original technical environment while adapters are used in order to connect them to the federation.

A federation model is not simply static, it possesses dynamic behaviors. The purpose of defining a federation is then twofold: first, it gathers descriptive parts reifying dependencies; second, it coordinates behaviors that use the reified dependencies. Note that a federation serves an intention (or purpose) which defines its objective (*e.g.*, keep the synchronization in a network of models) and thus not all dependencies need to be reified but only those that would serve its intention. Note as well that the federation model is itself a model and can be manipulated as such.

III. Approach

This section describes our approach to build *security-oriented* model federations that have as intention to support the consistent evolution of secured models. Our approach is independent of specific models and/or model-federation approaches. Nevertheless, in order to ease the discussion, we instantiate the description of our approach to our use case (discussed in Section IV).

The proposed approach focuses on the reification of dependencies between the participating models even when the models correspond to different modeling languages and are built using different design tools. With our approach, users can detect if the models remain consistent in the context of security after one or many changes on any model. To do so, users can define their own *security rules* depending on their needs and the requirements of the system. These security rules are attached to various links (which reify dependencies) according to their semantics.

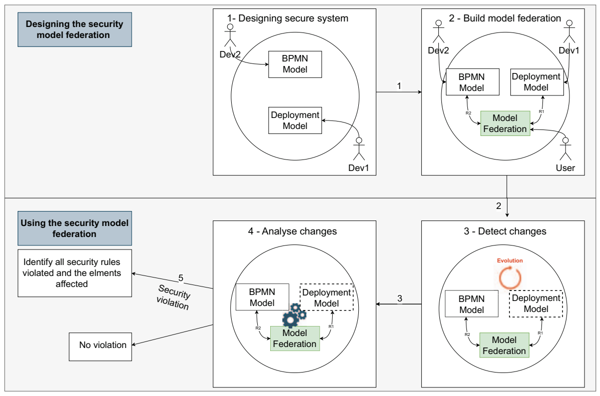


Fig. 1. Excerpt of *the security-oriented model federation* methodology

Our methodology involves two phases: (1) Designing the security model federation and (2) Using the security model federation to monitor evolution. Note that our methodology may be applied independently of the system development process. It can come before the actual system design, concurrently with it, or after a first version of the system is built.

Figure 1 describes the process of our methodology, how the model federation technique is adapted to link the system models and how it is applied. The rest of the section describes in more detail these steps.

A. *Designing the security model federation*

1) *System Design:* The objective of this step is to create the system models, including security aspects. For each model, the responsible development team first identify and analyze the system's security requirements throughout its life cycle. Next, they choose the appropriate meta-model, tool and approach that enables them to specify their needs and express security requirements in the early phases of the software design.
2) *Build the security model federation:* This stage is divided into three steps, with the global objective of linking together all the models by federating them in a single modeling environment. This is represented by the green box (Model Federation) in Figure 1.
    a) *Identify the dependencies:* While each type of meta-model describes a specific kind of model which describes a given aspect of the system, these meta-models remain independent in their own technological space. Nevertheless, the elements of each meta-model may be explicitly or implicitly dependent.

The challenge is to identify these dependencies. At this level, it is assumed that the teams designing the system are aware of the various meta-models which compose it and are able to identify dependencies manually. Notice that this step requires collaboration and may require security experts. Among all the dependencies that may be found, our methodology addresses only those related to security (directly or indirectly).

b) *Establish the links:* In this step, each identified dependency is refined by creating links between dependent modeling elements. Following, the model federation approach, each model remains independent and a new model, the security model federation, is created. The elements of this security model federation must be designed using the tool chosen to support the approach and then linked to the initial modeling elements. We give details about such a tool in Section IV. This step is a modeling step where the modeling elements reify links, and must follow the usual modeling process.

c) *Define security-oriented federation:* In order to be able to automatically detect inconsistencies, consistency must be first described in the form of security rules. These rules are written by using the facilities provided by the model-federation approach of choice (Openflexo [10] in our case). Security rules are then manually attached to the corresponding links that relate elements of the heterogeneous models so that when changes occur only the related rules need to be (re)evaluated.

B. Using the security model federation

We have presented above a methodology to build *Security-oriented* model federations. Its objective is to help in identifying inconsistencies related to security when the involved models evolve. To do so, these federations are used as follows.

1) *Detect changes:* The first step in the use of *security-oriented* model federations is the detection of model changes. This detection of changes may be done manually (*e.g.*, by the user performing the modifications) or automatically (*e.g.*, by leveraging the facilities of the selected approach to design and implement the model federation).
2) *Analyze changes:* First, we need to select the changed elements. Then, check whether these elements are related to other elements by reified links. When those links contain security rules, we evaluate them in order to identify those that are violated and report the detected inconsistency, including the rule and the impacted elements.

IV. Case study

To evaluate our methodology, we carry out a case study using the iTrust open-source system [11]. We select iTrust as our case study since it has been previously used in relevant research works dealing with security critical software/systems. We focus on its security requirements: privacy, integrity, authentication, authorization.

The iTrust Medical Application was launched by W. Laurie in 2005 at the University of North Carolina. The objective of this project is to assist software engineering students in handling a complex system adapted to the reality. This project enables them to understand the importance of security and privacy requirements [7]. It is a fully functional system with suitable documentation [14] including requirements specified. In the following we illustrate the application of the methodology described in Section III of the iTrust system.

A. Designing the security model federation for iTrust

1) *Design the iTrust system:* To model the iTrust system we rely on: available models in literature [7], the iTrust public documentation, and finally, the source code[1].
Among all the possible aspects and models of the systems, we focused on the following models: deployment model, data model (UML class diagram), access control model and BPMN model. We used the Papyrus tool to build the deployment model which shows how iTrust is installed in a hospital, and the Data model that defines the structure, organizes system information and actions to control data in the database of iTrust. Additionally, we used BMPN2 to model iTrust processes and a custom EMF access-control metamodel for authorization policies. To capture security concerns in the aforementioned models, for simplicity, we opt to define UML Profiles. These profiles allow us to annotate models with security requirements. We reuse the annotations described in UMLsec and secBMPN. In further versions of this case study, we will directly use UMLsec and secBMPN models so that we can also reuse their analysis tools.
2) *Build the security-oriented model federation*
   a) *Identify the dependencies:* We analyze the different meta-models to identify security related dependencies. For example, we had identified that Pool/Lane elements correspond to artifacts in a deployment model and thus are dependent. Some of these dependencies, the ones in which data is transferred between different nodes over a non secured channel, are later equipped with security rules.
   b) *Establish the links:* To establish links, we implement the model federation by relying on OpenFlexo, a mature model-federation framework. This framework provides us the infrastructure required to connect multiple models while keeping each model in its technology space. Openflexo [10], [15] offers a Federation Model Language (FML) engine with an integrated model design environment which permits the design of models and their behaviour. Also, offers some mature *technology adapters* to connect external models to Openflexo. Among others we can find connectors to BPMN, excel, XML, EMF and OWL.
   Figure 2 illustrates the structure of the *security-oriented* model federation created for the iTrust system. The main structural element in FML is the *Virtual Model* which may be assimilated to a model in other

[1]https://github.com/ncsu-csc326/iTrust2

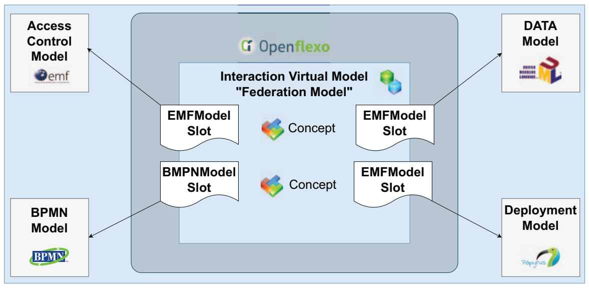


Fig. 2. Model federation with OpenFlexo

more classical frameworks. For our case study, we create a *Virtual Model* to serve as an interaction model, this is, a model capturing dependencies between model elements in different models. This allows us to establish links between elements in the models participating in our *security-oriented* federation (*e.g.*, BMPN and deployment models). Practically, this is achieved by the creation of *Flexo Concepts* (which can be seen as metaclasses) in the aforementioned interaction model. To enable these concepts to access information on external models (*e.g.*, models not directly defined in FML), *Openflexo* uses *model slots* which serve as technological connectors. For example, to link the Pool element of the BPMN model with the Artifact element of the Deployment model, we create a concept called ”PoolToArtifact” as shown in listing 1. Two technology adapters (one for BPMN models and one for Deployment models) are used.

```
public concept PoolToArtifact {
  public Pool poolInter with ConceptInstance(
     virtualModelInstance=BPMNModelSlot);
  public Artifact artifactInter with ConceptInstance
     (virtualModelInstance=deployModelSlot);
    String namePoolArtifact;
  public create::_createPoolArtifact(Pool srcPool,
     Artifact destinatArtifact) {
      poolInter = parameters.srcPool;
      artifactInter = parameters.destinatArtifact;
        container.listConceptBPMNDeploy.add(
     namePoolArtifact);
    }
  delete() {
      delete poolInter;
      delete artifactInter;
    }
  }
```

Listing 1. FlexoConcept linking Pool to Artifact with FML

c) *Define security-oriented federation:* Regarding the security requirements of the iTrust system, for now we have defined security rules concerning the transmission of data, who access it and how this relates to changes of deployment. Openflexo enables us to create a set of behaviors that are used to express security rules using the FML language. In our interaction model, each concept that represents a dependency between models contains a security rule or a set of security rules.

For example, we have attached the security rule in Listing 2 to the concept ”PoolToArtifact”. The objective of this security rule is to verify whether the communication flow between tasks have the same security annotation as the communication path relating Device/Node where the artifact is deployed.

```
public checkSecurityConsistency() {
    boolean isEncrypeted = true;
    if (!artifactInter.deployaccess.deviceDep.comPath.
     secNotationDeploy.equals("encrypted")) {
        isEncrypeted = false;
    }
    List<Lane> lanes = select Lane from poolInter;
    for (Lane lane : lanes) {
        List<Task> tasks = select Task from lane;
        for (Task task : tasks) {
            if (!task.comPath.secureNotation.equals("
     encrypted")) {
                isEncrypeted = false;
            }
        }
    }
    if (!isEncrypeted) {
        log "Security check failed";
    }
}
```

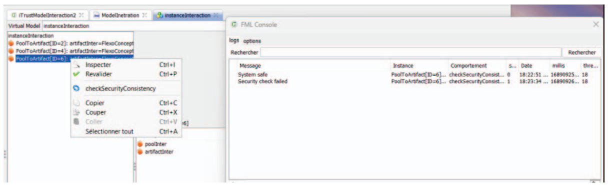
Listing 2. Security rule

B. *Using the iTrust security-oriented model federation*

In order to illustrate the use of the iTrust *security-oriented* model federation we describe here an evolution scenario related with a deployment change. Concretely, we propose an evolution scenario in which an artifact is moved from a trusted device to a non trusted one, which implies communications to occur through an encrypted channel (in UMLsec, a channel with an *encrypted* annotation).

1) *Detect changes:* At this stage, we detect changes manually. However, in future versions we will detect changes automatically by leveraging the functionality of the Openflexo framework, which allows us to define reactive behavior using the FML language.
2) *Analyze changes:* Once the change in the deployment model is detected, it needs to be analyzed. The security consistency corresponding to the change must be checked by using predicates. For example, artifacts of the deployment are linked to Pools and Lanes in the BPMN model, and a security consistency rule is attached to this link. This rule needs to be re-evaluated. The violation shown in Figure 3 is caused by the communication flow between Tasks with Pool that is not annotated ”encrypted”.

Summarizing, the *security-oriented* model federation for the iTrust system helped us to: 1) detect that changes occurred in the deployment model; 2) Analyze the change to determine its impact in other models and on the security of the system.



instanceInteraction
PoolToArtifact[ID=2]: artifactInter=FlexoConcept
PoolToArtifact[ID=4]: artifactInter=FlexoConcept
PoolToArtifact[ID=6]:
poolInter
artifactInter

```
Message               Instance                Comportement              s...  Date        millis         thre...
System safe           PoolToArtifact[ID=6]... checkSecurityConsist...  0     18:22:51 ... 16890925...   18
Security check failed PoolToArtifact[ID=6]... checkSecurityConsist...  1     18:23:34 ... 16890926...   18
```

Fig. 3. Security rule violated after deployment model evolution

V. Related work

Many different approaches have been proposed to support the security by design paradigm at the model level. Among others, we can find UMLsec [1], SysML-Sec [2], and secBPMN [3] which add security to UML, SysML and BMPN respectively. Our approach is complementary to these contributions when they are used in a multi-model scenario by dealing with their consistent evolution. In this sense, our approach fits in the feature model provided by [16] describing multi-model evolution approaches.

A plethora of works deal with consistency management in multi-model scenarios [17], some use the model federation paradigm as we propose in our approach. In [18], they define a concept meta-model to establish inter-model relations called Commonalities. Similarly, in [19] the authors propose to develop a framework that provides endpoint federation for existing *GraphQL WS-interfaces*. From a more theoretical point of view, in [20] the authors present *Comprehensive systems*, a category theory based structure in which multi-model systems can be described, demonstrating that existing consistency verification and repair approaches may be used over it. Security is not a concern in the aforementioned approaches, although they could certainly be adapted to deal with it, *e.g.*, by implementing our methodology. We choose Openflexo as our model federation infrastructure due to its flexibility and maturity.

Focused on evolution and security, in [9] Sven Peldzus propose an approach on top of UMLsec to study the evolution of system artifacts (conceptual level to program level) by using inter-artifact tracing and graph transformation techniques to synchronize models. Object Constraint Language (OCL) predicates are used to verify intra-model consistency w.r.t security. Similarly, in [1], [7], [8] the authors study the consistent evolution of the environment according to the security requirements of the system. They use SiLift to detect changes and a set of rules to semi-automatically restore system security. We believe that (parts of) their approach may be integrated into ours to enhance it. As an example, their SMRs may be used as behaviours in our links. Similar to our approach, albeit not dealing with evolution, in [21] the authors propose a model-based approach that enables the composition of heterogeneous artifacts into a consistent system model that can then be used for verification and simulation. Their approach is based in Transformation Rules Expressions (TRE) that are used to build a combination of functional and security views at the meta-model level. In contrast to their approach, we use model federation to establish relations between different modeling languages without the need for a combined metamodel.

VI. Conclusions & future work

We have presented here an approach to support the evolution of secured models (*i.e.*, models including security concerns) in a multi-model MBSE scenario by leveraging on the model federation paradigm. We have presented a methodology to create *security-oriented* model federations and applied it to a case study based on the iTrust system.

As a future work, we plan to enrich our approach by including: 1) additional types of models at different abstraction levels; 2) explore automation potential for some steps of our methodology; and 3) improve the validation and tool support.

References

[1] J. Bürger, S. Gärtner, T. Ruhroth, J. Zweihoff, J. Jürjens, and K. Schneider, "Restoring security of long-living systems by co-evolution," in *COMPSAC 2015*, vol. 2. IEEE, 2015, pp. 153–158.

[2] Y. Roudier and L. Apvrille, "Sysml-sec: A model driven approach for designing safe and secure systems," in *MODELSWARD*. IEEE, 2015, pp. 655–664.

[3] M. Salnitri, F. Dalpiaz, and P. Giorgini, "Designing secure business processes with secbpmn," *Software & Systems Modeling*, vol. 16, pp. 737–757, 2017.

[4] Z. K. Kebaili, D. E. Khelladi, M. Acher, and O. Barais, "Towards leveraging tests to identify impacts of metamodel and code co-evolution," in *CAiSE 2023*. Springer, 2023, pp. 129–137.

[5] M. Riedl-Ehrenleitner, A. Demuth, and A. Egyed, "Towards model-and-code consistency checking," in *COMPSAC*. IEEE, 2014, pp. 85–90.

[6] R. Jongeling, J. Fredriksson, F. Ciccozzi, A. Cicchetti, and J. Carlson, "Towards consistency checking between a system model and its implementation," in *ICSMM 2020*. Springer, 2020, pp. 30–39.

[7] J. Bürger, D. Strüber, S. Gärtner, T. Ruhroth, J. Jürjens, and K. Schneider, "A framework for semi-automated co-evolution of security knowledge and system models," *Journal of Systems and Software*, vol. 139, pp. 142–160, 2018.

[8] S. Peldszus, J. Bürger, T. Kehrer, and J. Jürjens, "Ontology-driven evolution of software security," *Data & Knowledge Engineering*, vol. 134, p. 101907, 2021.

[9] S. M. Peldszus, *Security Compliance in Model-driven Development of Software Systems in Presence of Long-Term Evolution and Variants*. Springer Nature, 2022.

[10] F. R. Golra, A. Beugnard, F. Dagnat, S. Guerin, and C. Guychard, "Addressing modularity for heterogeneous multi-model systems using model federation," in *Companion Proceedings of the 15th International Conference on Modularity*, 2016, pp. 206–211.

[11] A. Meneely, B. Smith, and L. Williams, "Appendix b: itrust electronic health care system case study," *Software and Systems Traceability*, p. 425, 2012.

[12] S. P. Mohanty, "Security and privacy by design is key in the internet of everything (ioe) era." *IEEE Consumer Electron. Mag.*, vol. 9, no. 2, pp. 4–5, 2020.

[13] F. R. Golra, A. Beugnard, F. Dagnat, S. Guerin, and C. Guychard, "Continuous requirements engineering using model federation," in *RE 2016*. IEEE, 2016, pp. 347–352.

[14] J. C. Maxwell and A. I. Antón, "Checking existing requirements for compliance with law using a production rule model," in *2009 second international workshop on requirements engineering and law*. IEEE, 2009, pp. 1–6.

[15] F. R. Golra, F. Dagnat, J. Souquieres, I. Sayar, and S. Guerin, "Bridging the gap between informal requirements and formal specifications using model federation," in *SEFM 2018*. Springer, 2018, pp. 54–69.

[16] P. Stünkel, H. König, A. Rutle, and Y. Lamo, "Multi-model evolution through model repair," 2021.

[17] W. Torres, M. G. Van den Brand, and A. Serebrenik, "A systematic literature review of cross-domain model consistency checking by model management tools," *Software and Systems Modeling*, pp. 1–20, 2020.

[18] H. Klare and J. Gleitze, "Commonalities for preserving consistency of multiple models," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 2019, pp. 371–378.

[19] P. Stünkel, O. von Bargen, A. Rutle, and Y. Lamo, "Graphql federation: A model-based approach," 2020.

[20] P. Stünkel, H. König, Y. Lamo, and A. Rutle, "Comprehensive systems: a formal foundation for multi-model consistency management," *Formal Aspects of Computing*, vol. 33, no. 6, pp. 1067–1114, 2021.

[21] H. Zhao, F. Mallet, and L. Apvrille, "A language-based multi-view approach for combining functional and security models," in *APSEC 2019*. IEEE, 2019, pp. 426–433.