D. Ferraris, C. Fernandez-Gago, and J. Lopez, "Verification and Validation Methods for a Trust-by-Design Framework for the IoT", *36th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec'22)* vol. 13383, pp. 183-194, 2022.
http://doi.org/https://doi.org/10.1007/978-3-031-10684-2_11
NICS Lab. Publications: https://www.nics.uma.es/publications

Verification and Validation Methods for a Trust-by-Design Framework for the IoT

Davide Ferraris Carmen Fernandez-Gago Javier Lopez

July 28, 2022

Abstract

The development of an Internet of Things (IoT) entity is a difficult process that can be performed following a System Development Life Cycle (SDLC). Two important phases of a SDLC process are verification and validation (V&V). Moreover, if we want to guarantee that trust is considered through the SDLC we have to implement it since the first phases and verify and validate its implementation during V&V. Verification usually is defined as "the system has been built right", on the other hand validation refers to the fact that "the right system has been built". Concerning trust, following our methodologies we can state that we can verify that "the trusted IoT entity has been built" and validate that "the right trusted IoT entity has been built". In this paper, we propose a methodology to verify and validate requirements related to a trusted IoT entity. Following the methodology, it is possible to check if the requirements elicited in the early phases of the SDLC have been implemented in the developed functionalities. These final phases will be fundamental in order to achieve trust in the developed IoT entity.

Keywords. Trust, SysML, UML, Internet of Things (IoT), System Development Life Cycle (SDLC).

1 Introduction

The Internet of Things (IoT) allows humans and smart entities to cooperate among them anyhow and anywhere [23]. The IoT entities are growing each year and "there are expected to be more than 64B IoT devices worldwide by 2025" [1]. This prediction states that that the IoT paradigm will define how the world will be connected. For this reason, even if there will be more opportunities to be connected, also many issues will arise. We believe that security and trust can help to solve these issues implementing them in the IoT during the System Development Life Cycle (SDLC) [12]. Usually, the SDLC is composed of different phases. In the earliest ones, the requirements are elicited. This phase

[1] https://techjury.net/blog/internet-of-things-statistics/

will lead to the development of the entity, but in order to conclude the process it is fundamental to verify and validate the requirements. In fact, through verification is possible to say that *the entity has been built in the right way* that means that the functionalities are working as expected. On the other hand, validation means that *the right entity has been built*. In this case, we can say that the IoT entity has been developed as it was intended for the originated need. Moreover, in the SDLC of an IoT entity, the developers must tackle other challenging tasks. One is how to consider the dynamic environment where the IoT device will interact with other devices. We believe that trust can help overcome this issue, because a trusted device will ensure that the interaction can be performed in a secure way. Moreover, considering trust during the whole SDLC will prevent later issues and as demonstrate by [15], when a task is conducted early in the SDLC, it will save money for the whole project instead of consider it later [19]. However, trust is hard to define because it is strongly dependent on the topic in which is considered [9]. Moreover, trust is connected to the context, in fact it "means many things to many people" [7]. Nevertheless, usually at least two actors must be involved in a trust relationship: the trustor and the trustee. The trustor is the one that needs the trustee in order to fulfill an action or a service. For example, in an IoT environment the trustor can be the user and the trustee can be the IoT device. In this paper, we will propose a method that help developers in verifying and validating requirements related to a trusted IoT entity.

The structure of the paper is as follows. In Section 2, we explain the previous work connected to this paper. Then, in Section 3, we describe the related work about verification and validation. Then, the core of this paper concerning requirements verification and validation phases are presented in Section 4 and 5. The implementation of these phases are explained in Section 6 and 7. Finally, in Section 8, we conclude and discuss the future work.

2 Background

IoT allows smart entities to be connected through the Internet. This configuration brings many possibilities (i.e., to connect entities even if they are far). On the other hand, the connection among them can trust or security issues. Another difficulty that is strongly dependent on the IoT paradigm is that the smart entities are usually produced by different vendors and they cover different topics [20]. This variety increase the issues but also the potentialities of the IoT [5].

In our previous works, we have developed a framework that considers trust holistically during the whole SDLC of a smart IoT entity[12]. The framework is composed of a K-Model shown in Figure 1 and several transversal activities (i.e., Traceability, Risk Analysis). Moreover, it is fundamental to take the context into consideration during each phase of the SDLC. This aspect is very important for IoT due to the dynamicity and heterogeneity of this paradigm. The context can depend on several aspects such as the environment or the functionalities of

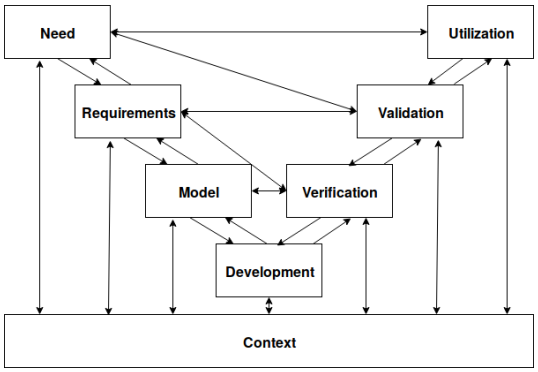

Figure 1: K-Model: with the *Transversal Activities* it compounds the framework [12]

an entity.

In the K-Model, we can see different phases covering all the SDLC of the IoT entity under development: from cradle to grave. The first phase is related to the needs phase, where it is proposed the purpose of the new IoT product. All the stakeholders related to the entity have a key role on it. The second phase considers the requirements elicitation process. In this phase, the developers must elicit the requirements according to the needs considered in the previous phase. About this part, we proposed TrUStAPIS [11]. This methodology consider trust according to other domains such as security, usability, privacy, availability, safety, identity. We believe that considering other domains according to trust is necessary to guarantee it in a virtuous circle that enhance the protection of the IoT entity under development. To state this assumption, we have followed and expanded Hoffman and Pavlidis' works [16, 21].

The output of the requirements elicitation process must be taken into consideration when developers will perform verification and validation in latter phases of the K-Model. In the middle, there is the third phase which considers the model definition [22]. In fact, it is fundamental to refine the requirements turning them into models useful for the developers in order to realize the IoT entity in the following and central phase of the K-Model that is the development phase. Thus, before the entity is delivered two important phases, the core of this paper, must be performed: verification and validation. Verification phase is connected to Requirements and Model phase as shown in Figure 1. On the other hand, Validation is connected to Requirements and Need phase. This difference is very important in order to perform these activities. For both the phases, we will use JSON code developed in the second phase (i.e., Requirements) of the K-Model, in order to help the developers to perform both the verification and validation process.

JSON has been chosen because "it is easily readable by humans and machines" and as stated in [2] this aspect allows developers to share it among

either stakeholders and applications. JSON is also useful to connect the requirements and the needs in the validation phase as we will show in Section 5. However, Verification and Validation has been considered during the years. In the next section, we will briefly introduce them.

3 Related Work

Verification and, later in the SDLC, validation are two phases performed to prove that the entity has been properly developed. If validation is related to the final product, on the other hand, verification can be performed in any phase in order to find errors or modify the original plan. Anyhow, it is important to consider verification also as a separate phase, in order to check completely that the functionalities of the IoT entity work as planned. A definition of verification appears in the Project Management Body of Knowledge (PMBOK) [6] where verification is considered as "the evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with validation". This phase is fundamental in order to verify that *the entity has been built in the right way*. This means that all the specifications have been followed in a correct way. In the verification phase the functionalities of the entity are tested as well as its correct implementation. However, tests are performed both in verification and validation. In fact, Linhares et al. [18] stated that "test is realized on the implemented system". It involves a set of possible inputs executing the functionalities in order to check if something wrong occurs. Another element that is used during verification processes are the inspections. Fagan et al. [8] described for the first time the inspection processes which were lately considered by Ackerman et al. [1] that described the inspection processes as fundamental for the verification phase. However, it is possible that the developed entity has passed the verification phase, but it fails when validated. This situation can happen when the entity has been built for the specifications, but the specifications themselves fail to address the user's needs. Validation is defined by the PMBOK [6] as "the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with verification". By validation we state that *the right entity has been built* [15]. This means that the desired entity has been developed as the users and vendors wanted to be. In addition, this phase checks that the entity works properly in a real system's environment. In order to understand the problem, there can be different categories of validations. Process validation has been analyzed in [17]. This guidance promotes a "lifecycle" approach. Then, retrospective validation is executed for a product that is already distributed or sold to the customers [14]. Next, partial validation is often used for research purposes or prototypal studies if the time before the utilization phase is constrained. Finally, a powerful tool that can be used is called *Independent Verification and Validation* (IV&V), as stated by Arthur et al. [4]. This is a tool that can be exploited in order to mitigate the growing complexity

related to the expansion of modeling and simulation problems. Moreover, in a later work [3], the author showed an important issue with IV&V according to agile development. One of them was related to traceability and documentation. However, with our work, we have implemented traceability that allows developers to solve the aforementioned issues.

4 Verification

In this paper, we expand and explore the fifth and sixth phases of the K-Model [12] where we have only proposed them without any further explanation. Thus, the verification process is keen to analyze the requirements according to the models and to test the functionalities in order to prove that the requirements are well formed and the "system has been built right". At the end of this phase we can state that "the trusted IoT entity has been built right". The verification process is the following. The steps are shown in Figure 2.

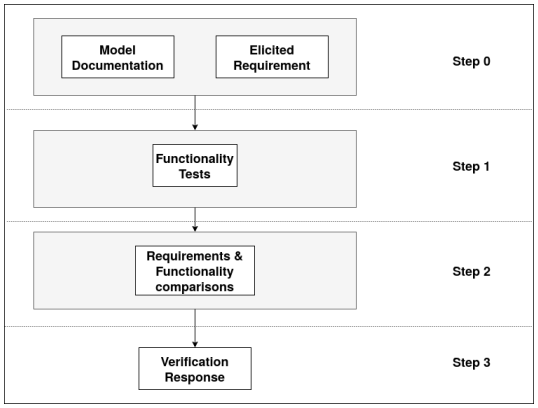

Figure 2: Verification Process: step-by-step methodology

The output of each step is the input for the following steps.

1. The step zero is related to the comparison of the originating model documentation (collected during the third phase of the K-Model) and the elicited requirements. In fact, we need to compare the requirements and the models in order to verify the functionalities of the developed IoT entity as they have been modeled and planned.

2. The first step is performed in order to execute tests on the functionalities.

3. In order to check that the requirements have been respected, the functionalities tested in the previous step must be connected to the originating requirements to certify that they have been designed in a correct way.

4. The response of the comparison performed in step 2 is the output of the final step. If the functionality follows the requirements, then, the verification test is successful, otherwise the functionality must be modified according to the elicited requirements. This feedback is represented in the K-Model and it is necessary in order to develop the correct IoT entity.

5 Validation

The validation process is based on the fact that "the right system has been built". In order to clarify this, we need to check that the IoT entity and its functionalities match the initial stakeholder's need. Each need is delineated by a statement and it is represented by one or more stakeholders. The stakeholders are the "persons" which have an interest in the system. These needs must be collected by the developers in the first phase of the K-Model. Then, in the requirements phase, the needs are analyzed and transformed in requirements. We resumed these two phases because they are considered also in the validation phase to check if the IoT entity represents the original needs. At the end of this phase, we can state that "the right trusted IoT entity has been built". The validation process is the following. The steps are shown in Figure 3.

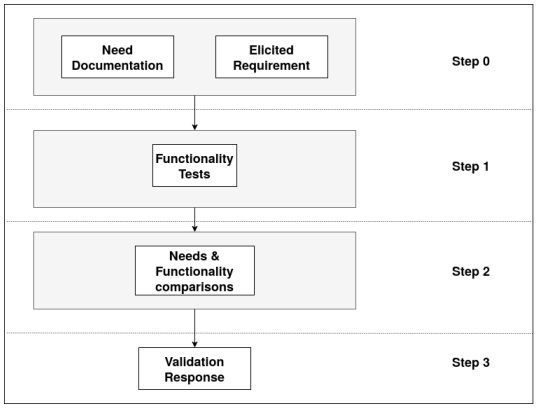

Figure 3: Validation process: step-by-step methodology

The output of each step is important in the following steps.

1. The step zero is related to the comparison of the originating needs and the elicited requirements. In this phase, the needs and the connected requirements are compared, this step is helpful to discover if there are some missing or wrong requirements.

2. The first step is performed in order to execute tests related to the functionalities developed following the elicited requirements satisfying the needs.

3. The comparisons between needs and requirements is performed in the second step, where the output of the test is compared to the originating need.

4. The response of the comparison performed in the previous step is considered in the last step of the methodology. If the functionality satisfies the need, the validation test is successful, otherwise the functionality must be modified. This means that the developers must go back in the K-Model to the previous phases in order to perform the correct modifications. Anyway, all the previous phases, if correctly performed, reduce this possibility.

6 Verification Scenario: Smart Cake Machine

In this section, we will present how to verify the requirements presented in [11] concerning a use case scenario regarding a Smart Cake Machine (SCM) previously introduced in [12]. We briefly describe here the scenario for the readers. Let us assume that, after a market analysis, the stakeholders have had the idea to produce a SCM. It should tell the users which ingredients are needed for a particular cake where the recipes can be downloadable from a website or inserted by the users. Moreover, it is supposed that this device can communicate with other IoT entities. In this case, let us assume that the SCM can check the Smart Fridge (SF) for an ingredient or it can send a request to the trusted nearest supermarket (SM) in order to buy missing ingredients. This operation must be performed through a Smart Hub (SH) in order to preserve the SCM from possible external attacks [10]. According to the users, the SCM must check the trusted users allowed to interact with. From these needs and considering the context, the requirements are elicited following TrUStAPIS methodology [11] and the models following [13], Then, the functionalities have been developed in the central phase of the K-Model. Now, there are verification and validation phases to be performed. In this section, we present the verification.

Step 0. The elicited requirements related to [11] are shown in Table 1. Then, in the third column, we can see the model connected to the requirement. These models have been presented in [13]. In this step, we compare the model documentation and the elicited requirements.

```
"IoT_requirement_USAB01" : {
  "Context" : {
    "Domain" : [{
      "Usability" : {
        "Characteristic" : ["Simplicity, Understandability"] } }],
    "Environment" : "Smart Home",
    "Scope" : "User Interface" } ,
  "Actor" : {
    "Role" : "User",
    "Type" : ["Human User"] },
  "Action" : {
    "Type" : ["Fulfill"] ,
    "Measure" : [" "]},
  "Goal" : "Let the user insert new recipes"
},
```

Figure 4: JSON code for USAB01

Table 1: Requirements and model connections.

Domain	Requirement	Model
Usability	**USAB01** - The user shall be able to insert new recipes	**State Machine Diagram 1 (SMD1)**: Upload a recipe into the SCM
Privacy	**PRIV03** - User data shall be kept private	**Activity Diagram 1 (AD1)**: Securing user data
Security	**SEC02** - The SCM shall delegate the Smart Hub to order the missing ingredients	**Sequence Diagram 1 (SD1)**: User, SCM, SF, SH and SM interaction
Identity	**IDNT02** - The user shall provide his/her data in order to be registered	**Use Case Diagram 1 (UCD1)**: User data
Trust	**TRST01** - The SCM shall trust a Smart Supermarket with a trust level above 0.5	**Class Diagram 1 (CD1)**: SCM and SM service classes

In this example, we will show only how to verify requirement USAB01 according to SMD1 diagram. Thus, in Figure 4, we show the JSON related to USAB01. In fact, it is useful to speed up the verification process by checking if the goal and the characteristics are met.

Step 1. After the collection of the requirements and the models performed in the previous steps, the developers must perform the functionality tests. In this case, it is necessary to verify different aspects. In first instance, that only a trusted user can access the SCM. In fact, only after this important step, it is possible to insert a new recipe. Then, if the first control is positive, the functionality tests must check if the recipes are correctly uploaded. SMD1 was designed in the following way. Firstly, for any new recipe, it was mandatory to insert a new title. Secondly, the items and their quantity. The recipe upload ends when all the items have been inserted. Thus, the test must check if all the items are correctly inserted. If not, the test fails, and the code must be checked in order to find the error. Otherwise, the functionality test passes, and it is possible to proceed to the following step.

Step 2. This step is a second check that guarantees that the requirements are met, and the functionalities have been correctly implemented. Traceability guarantees this aspect as we have seen in Table 1. In the case the previous step has ended correctly, it is possible to confirm that the requirement USAB01 and its derived functionalities have been corrected implemented. On the other hand, it will be raised an issue that will be checked in order to adjust the wrong functionality. After all the functionalities have been checked positively or negatively, it is possible to proceed to the final step of the verification process.

Step 3. The verification response is positive if all the previous steps have been correctly performed. Thus, a document certifying which tests have been positively or negatively performed will be the output of this phase. In the case

some tests have been negatively performed, there will be a modification of the functionalities, and a new verification process will be performed after the needed modifications.

7 Validation Scenario: Smart Cake Machine

In this section, we will present how to validate the SCM. In the validation phase, the tests are performed in the environment, in order to check if the needs are met and the cooperation with other IoT entities is fulfilled. In Figure 5, it is possible to see the connections among the SCM and other IoT entities.

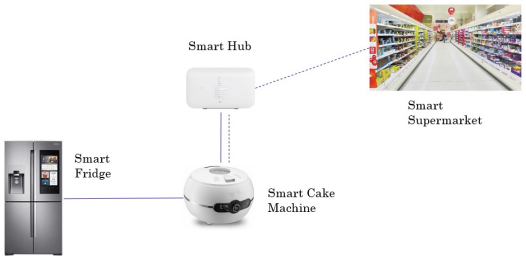

Figure 5: Smart Cake Machine and its relationships with the other IoT entities

Step 0. In this step, it is performed the comparison of the *Need* documentation and elicited requirements. According to [11], we have identified the following subset of originating needs:

1. **Need 1**: The temperature of the SCM must be checked and it could not overcome 250°C.
2. **Need 2**: The recipes must be downloadable from the vendor website or inserted by authenticated users. Authentication must be performed by code.
3. **Need 3**: The SCM could interact with a SF to check if a particular ingredient is present. If not, the SCM can interact with a trusted SM through the home SH and order the missing ingredient.
4. **Need 4**: The communication among the smart home entities must be guaranteed and encrypted.

In the third column of Table 2, we can see the need connected to requirements. Needs X are not strictly connected to the requirements, but to the context.

For the validation test, we consider requirement TRST01 and Need 3.

Step 1. As we will present in the following and final phase of the K-Model, an algorithm is needed to introduce the new IoT entity in a smart home

Table 2: Requirements elicited using TrUStAPIS from [11]

Requirement	Statement of the Requirement	Need
Trust Req.	**TRST01** - The SCM shall trust a Smart Supermarket with a trust level above 0.5	**Need 3**
Usability Req.	**USAB01** - The user shall be able to insert new recipes	**Need 2**
Security Req.	**SEC01** - The user shall be authenticated	**Need 2**
Security Req.	**SEC02** - The SCM shall delegate the Smart Hub to order the missing ingredients	**Need X**
Availability Req.	**AVBT01** - The SCM shall be able to connect to the Smart Hub	**Need X**
Privacy Req.	**PRIV01** - The SCM shall perform an encrypted communication with the Smart Fridge	**Need 4**
Identity Req.	**IDNT01** - The user shall be authenticated	**Need 2**
Identity Req.	**IDNT01**.2 - The user shall be authenticated by code	**Need 2**
Safety Req.	**SFT01** - The SCM shall be able to check its temperature level	**Need 1**
Safety Req.	**SFT01.1** - The SCM temperature level shall be lower than 250°C	**Need 1**

context. Anyhow, in this case, we need to focus only on the functionalities that must be validated. In this case, it is related to the interaction between the SCM and a trusted SM. The trust value is provided by the SH or from the user (i.e., because the user has a direct past relationship with an SM). The value can be computed after considering multiple parameters (i.e., cost, distance, quality), and the service is satisfied if any interaction among SCM and the SM is performed if and only if the computed trust value is more than 0.5. If the interaction is performed at a lower level, the validation test fails. Otherwise, it succeeds.

Step 2. This step is fundamental in order to certify that the tested functionalities satisfy the requirements and fulfil the originating need. If the previous step ends correctly, it confirms that the originating need is met. Otherwise, an issue is raised, and the developers will need to check and correct it. After all the functionalities have been checked in their environment and compared with the needs, it is possible to continue to the final step of the validation process.

Step 3. The validation response is positive if all the previous steps have been positively performed. Thus, documentation is produced to certify that stakeholders' needs have been met and it is possible to proceed to K-Model's final phase: Utilization. However, if validation tests have failed, it will be produced documentation about the failed functionalities to discuss and fix the raised issues.

8 Conclusion and Future Work

Trust is important in the IoT environment. In order to guarantee trust in the IoT, we believe that it is useful to consider it during the whole SDLC. In this paper, we focused on two phases of the SDLC: verification and validation. In the verification phase, developers analyze the previously elicited requirements. Then, the functionalities are tested to prove that the entity has been built in the right way. Finally, during the validation process, the developers must check that the complete IoT entity matches its originating needs. Thus, if the validation process is correctly performed, it is possible to assert that the right IoT entity has been built. As a future work, we will perform the tests in a complex scenario.

Acknowledgement

This work has been supported by the Spanish Ministry of Science and Innovation Project SecureEDGE (PID2019-110565RB-I00) and by the EU H2020-SU-ICT-03-2018 Project No. 830929 CyberSec4Europe (cybersec4europe.eu). Moreover, we thank Huawei Technology for their support. This work reflects only the authors view and the Research Executive Agency is not responsible for any use that may be made of the information it contains.

References

[1] Ackerman, A.F., Buchwald, L.S., Lewski, F.H.: Software inspections: an effective verification process. IEEE software 6(3), 31–36 (1989)

[2] Alonso-Nogueira, A., Estévez-Fernández, H., García, I.: Jrem: an approach for formalising models in the requirements phase with json and nosql databases. International Journal of Computer and Information Engineering 11(3), 353–358 (2017)

[3] Arthur, J.D., Dabney, J.B.: Applying standard independent verification and validation (iv&v) techniques within an agile framework: Is there a compatibility issue? In: 2017 Annual IEEE International Systems Conference (SysCon). pp. 1–5. IEEE (2017)

[4] Arthur, J.D., Nance, R.E.: Independent verification and validation: a missing link in simulation methodology? In: Proceedings Winter Simulation Conference. pp. 230–236. IEEE (1996)

[5] Čolaković, A., Hadžialić, M.: Internet of things (iot): A review of enabling technologies, challenges, and open research issues. Computer Networks 144, 17–39 (2018)

[6] Edition, F.: Ieee guide-adoption of the project management institute (pmi®) standard a guide to the project management body of knowledge (pmbok® guide) (2011)

[7] Erickson, J.: Trust metrics. In: Collaborative Technologies and Systems, 2009. CTS'09. International Symposium on. pp. 93–97. IEEE (2009)

[8] Fagan, M.: Design and code inspections to reduce errors in program development. In: Software pioneers, pp. 575–607. Springer (2002)

[9] Fernandez-Gago, C., Moyano, F., Lopez, J.: Modelling trust dynamics in the internet of things. Information Sciences 396, 72–82 (2017)

[10] Ferraris, D., Daniel, J., Fernandez-Gago, C., Lopez, J.: A segregated architecture for a trust-based network of internet of things. In: 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC) (CCNC 2019). Las Vegas, USA (Jan 2019)

[11] Ferraris, D., Fernandez-Gago, C.: Trustapis: a trust requirements elicitation method for iot. International Journal of Information Security pp. 1–17 (2019)

[12] Ferraris, D., Fernandez-Gago, C., Lopez, J.: A trust by design framework for the internet of things. In: NTMS'2018 - Security Track (NTMS 2018 Security Track). Paris, France (Feb 2018)

[13] Ferraris, D., Fernandez-Gago, C., Lopez, J.: A model-driven approach to ensure trust in the iot. Human-centric Computing and Information Sciences 10(1), 1–33 (2020)

[14] Food, U., Administration, D., et al.: Guideline on general principles of process validation. US FDA: Rockville, MD (1987)

[15] Haskins, C., Forsberg, K., Krueger, M., Walden, D., Hamelin, D.: Systems engineering handbook. In: INCOSE, (2006)

[16] Hoffman, L.J., Lawson-Jenkins, K., Blum, J.: Trust beyond security: an expanded trust model. Communications of the ACM 49(7), 94–101 (2006)

[17] Katz, P., Campbell, C.: Fda 2011 process validation guidance: Process validation revisited. Journal of GXP Compliance 16(4), 18 (2012)

[18] Linhares, M.V., de Oliveira, R.S., Farines, J.M., Vernadat, F.: Introducing the modeling and verification process in sysml. In: 2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007). pp. 344–351. IEEE (2007)

[19] Marche, C., Nitti, M.: Can we trust trust management systems? IoT 3(2), 262–272 (2022)

[20] Nkuba, C.K., Kim, S., Dietrich, S., Lee, H.: Riding the iot wave with vfuzz: Discovering security flaws in smart homes. IEEE Access 10, 1775–1789 (2021)

[21] Pavlidis, M.: Designing for trust. In: CAiSE (Doctoral Consortium). pp. 3–14 (2011)

[22] Ponsard, C., Ramon, V.: Survey of automation practices in model-driven development and operations. Tech. rep., EasyChair (2022)

[23] Roman, R., Najera, P., Lopez, J.: Securing the internet of things. Computer 44(9), 51–58 (2011)