Novel Approaches for the Development of Trusted IoT Entities

Davide Ferraris   Carmen Fernandez-Gago   Javier Lopez

June 19, 2022

Abstract

The Internet of Things (IoT) is a paradigm allowing humans and smart entities to be interconnected anyhow and anywhere. Trust is fundamental in order to allow communication among these actors. In order to guarantee trust in an IoT entity, we believe that it must be considered during the whole System Development Life Cycle (SDLC). Anyhow, we think that usual development techniques are not effective for the IoT. For this reason, in this paper, we describe a methodology to develop an IoT entity by proposing a holistic approach implementing three different techniques: a bottom-up approach, a top-down approach and a trusted block development. Firstly, the top-down approach will start from the general IoT entity going down to its specific functionalities. Secondly, the bottom-up approach will focus on the contexts related to the IoT entity. It starts from basic ones, going up aggregating them to the composition of the IoT entity as a whole. Finally, the trusted block development will define different blocks of code related to functionalities and contexts. Every block can be considered a *trust island* where the contexts and functionalities are specified only for a particular block.

Keywords. Trust, SysML, UML, Internet of Things (IoT), System Development Life Cycle (SDLC).[1]

[1]

Davide Ferraris DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF MALAGA, MALAGA, 29010
E-mail address: ferraris@lcc.uma.es

Carmen Fernandez-Gago (Department of Applied Mathematics, Malaga, Spain, 29010
E-mail address: mcgago@lcc.uma.es

Javier Lopez DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF MALAGA, MALAGA, 29010
E-mail address: jlmlcc.uma.es

1 Introduction

The Internet of Things (IoT) is composed of two words: Internet and things. With these two words, we can understand the scope of this technology allowing the connection of things among them through the internet [19]. The word "thing" is generic and it can represent either humans or objects. In fact, through IoT, we can connect different types of things and how to connect them in a protected and trusted way is one of the main challenges in this field. In this paper, we will use the terms things, devices or entities for the same purpose.

Statista has predicted that 75.4 billions of devices would be connected by 2025[2]. This prediction is helpful to understand that the IoT paradigm will grow up to define how the world will be connected in the next years. For this reason, many opportunities will arise, but also many problems, especially related to trust and security [2]. A way to mitigate them is offered by security and trust.

Trust is difficult to define because it can be related to many different topics, from Psychology to Computer Science [5]. Moreover, trust is strongly connected to the context. In fact it "means many things to many people" [3]. In a trust relationship, there are basically two actors involved: the trustor and the trustee. The trustor is the one who needs a favor or a service, but he/she cannot fulfill it alone. For this reason, the trustor needs the trustee, which is the one keeping the trust. The level of trust between trustor and trustee can change over time positively or negatively due to the good or bad behavior of the trustee and on the outcome of the trust relationship. Moreover, it is strongly dependent on a particular context (i.e., a trustor can trust a trustee as a driver, but not as a cook).

Anyhow, the development of trust models for an IoT entity is a task that needs to be tackled. We believe that the usual System and Software Development Life Cycle (SDLC) approaches might be of help. For this reason, in this paper, we will focus on an important phase of the SDLC considering the central phase of the K-Model proposed in [8]: the development phase.

During the development of an IoT entity, the developers can consider several approaches in order to perform this crucial task. A widely used approach is the so-called top-down. It is basically a way to consider the problem starting from a general perspective to a specific one. Moreover, the top-down approach can be used even for software development through a Functional Breakdown Structure (FBS) or a Work Breakdown Structure (WBS) [13]. However, in our case, it is important to consider not only the functionalities but also their connections to the domains such as trust and security in order to divide and perform the analysis according to their scope.

On the contrary, the bottom-up approach starts from a specific viewpoint to a generic overview of the system. It is a method used particularly in software engineering [10, 13], but it can also be used to develop IoT infrastructures [18]. In our paper, we move forward and consider it according to the different contexts and domains of the IoT entities.

[2]https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

However, in our opinion, these two approaches alone are not enough for the development of a trusted IoT entity. In fact, it is important to highlight that an IoT entity is composed of software and an effective way to develop the code is following a finite state approach as stated by [22] and [1] creating trusted block of codes. This approach is even more important in an environment such as the IoT, where usually all the functionalities are performed separately and following a step-by-step process.

Nevertheless, top-down and bottom-up approaches have always been considered separately, such as the block or finite-state approach. In this work, we will extend these approaches by developing the block approach and implementing it to balance and connect top-down and bottom-up approaches among them. Moreover, the utilization of the block approaches will allow the developers to focus separately on the different contexts and functionalities according to the domain and their composition.

The structure of the paper is as follows. In Section 2, we describe the related work and the background of this paper. Then, the approaches designed for the development phase are presented in Section 3. In Section 4, we show how these approaches must be implemented in a use case. Finally, in Section 5, we conclude the paper and discuss the future work.

2 Related Work

In this section, we firstly discuss about IoT, trust and development techniques. Finally, we discuss about our previous work, showing why it is necessary to propose new development approaches in order to guide developers through the SDLC.

2.1 Trust, IoT and Development

In Information Technology, trust is strongly dependent on other domains such as security and privacy [12, 17]. Moreover, Ferraris et al. [8] stated that these relations are even more important for the IoT.

The IoT allows smart entities to be controlled anywhere and anyhow [19]. However, they must be secured through the Internet and it is essential to guarantee trust during the communication among smart entities [24]. In fact, we consider trust as “the personal, unique and temporal expectation that a trustor places on a trustee regarding the outcome of an interaction between them” [15]. According to this definition, we understand that there are at least two actors in a trust relationship: a trustor and a trustee. For this reason, we can state that there must be at least a trustor and a trustee in any trusted IoT communication.

Moreover, the IoT is a heterogeneous and dynamic field. In order to implement trust and its related domain in the IoT, it is necessary to consider it through the whole SDLC [8].

However, in the IoT the usual development techniques could not work properly also because of the actual network infrastructure, which was not built precisely for the IoT [23]. Solutions similar to the Software-Defined Networking (SDN) [4] can help to solve this problem allowing the development of new applications and techniques for the IoT [20]. Anyhow, if we can state that SDN is related to the network infrastructure, Software Defined Perimeter (SDP)[3] is related to the services and applications running on it. They can be used together to improve the functionalities of the IoT technology [4]. Thus, during the development of an IoT entity, it is helpful to take these infrastructures into consideration and to use development techniques that help to separate services, functionalities and contexts. These techniques can be represented by known development approaches such as the top-down and bottom-up approaches.

Ganchev et al. [11] proposed a top-down approach to create an IoT architecture for smart cities. Patel et al. [16] introduced a general approach that can be used to reunite the physical and the digital world belonging to the IoT paradigm. On the contrary, Reaidy et al. proposed a bottom-up approach to develop IoT infrastructures [18].

These works are interesting, but we need something more specific for a general IoT environment that also considers trust and related domains. In fact, as stated by [14] "an important question is what consequences a bottom-up and top-down construction of the IoT infrastructure has for the security, privacy and trust". Moreover, according to [21], it is important to find a balance between top-down planning and bottom-up innovation. We believe that in order to perform this task, it is necessary to implement a third approach that can consider both the approaches highlighting their points of strength and establishing a balance among them: a trusted block development. Thus, for these reasons, we develop the approaches that we will show in Section 3, considering them together to provide developers with a trusted block development technique based on the idea stated in [22, 1].

2.2 Background

In our previous works, we have developed a framework that holistically considers trust during the whole SDLC of a smart IoT entity [8]. The framework is composed of a K-Model and several transversal activities (i.e., Traceability, Risk Analysis). In addition, 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 depends on several aspects such as the environment, the functionalities of an entity or the rules of the company developing the IoT entity.

In the K-Model that is shown in Figure 1, 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 the purpose of the IoT entity to be developed is proposed the purpose of the IoT entity to be developed. In this phase, all the stakeholders related to the IoT device have a key role. The

[3]https://cloudsecurityalliance.org/working-groups/software-defined-perimeter

[4]http://www.waverleylabs.com/software-defined-network-sdn-or-software-defined-perimeter-sdp-whats-the-difference/

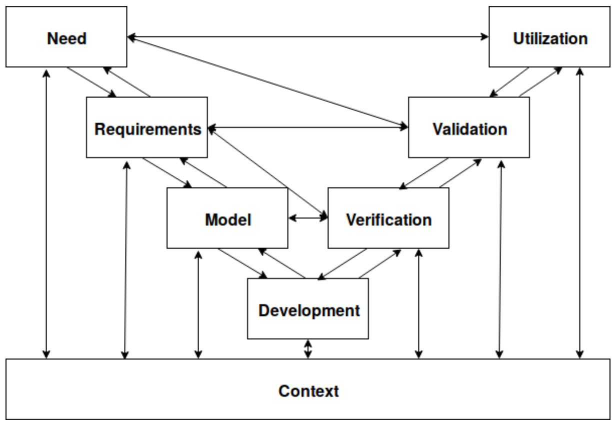


Figure 1: K-Model: with the *Transversal Activities* it compounds the framework [8]

second phase considers the requirements elicitation process. In this phase, the developers must elicit the requirements according to the needs considered in the previous one. We have described this phase in [7]. Then, the third phase considers the model specification. We have developed a model-driven approach to ensure trust in the modeling phase of the K-Model proposing several diagrams (i.e., requirement diagram, context diagram) to cover all the crucial aspects of the modeling phase [9]. The output of this phase is the input of the following and central phase of the K-Model. In this paper, we will consider, explore and expand it: the development phase.

3 Development

The development is the core phase of the K-Model presented in Section 2.2. This phase transforms the needs, requirements and models in the product that will be verified and validated in order to be distributed to the customers. In the IoT, the challenges are numerous because of the dynamicity and heterogeneity of this technology. The SDN could solve problems related to the old infrastructure and also using the SDP technology it is possible to create separated contexts to make the IoT entities communicate in a trusted and secure way. Ferraris et al. [6] proposed an architecture similar to the SDP technique that is useful to define the boundaries in a Smart home environment guaranteeing a trusted interaction among IoT entities. We will follow this idea and consider the possible contexts in which the IoT entity will be involved in order to develop it properly.

We believe that in order to holistically consider all the information collected in the previous phases of the K-Model, the developers must organize their work in a schematic way. Our proposed approaches help them fulfilling this goal. In fact, the top-down approach presented in Section 3.1 allows developers to consider functionalities firstly from a generic perspective and then in a specific way. On the contrary, the bottom-up approach discussed in Section 3.2 considers contexts from a specif point of view to a more generic one. The utilization of both approaches helps developers to better consider all the fundamental aspects of the IoT entity. Finally, both the approaches are considered during the trusted block development proposed in Section 3.3.

In order to show the steps of the development phase, in Section 3.4, we will present the order of utilization of our proposed approaches.

3.1 Top-Down approach

During the development of an IoT entity, a useful methodology is to analyze the IoT entity under development following a top-down approach.

In Section 2, we mentioned several methodologies implementing this approach (i.e., WBS or FBS). Moreover, in the previous phases of the K-Model, we have focused on how important it is to consider the domains related to trust. For this reason, we think that it is useful to mix the FBS considering also the related domains of each functionality.

We named this top-down approach FDBS (Functional Domain Breakdown Structure). In this approach, it is fundamental to highlight each functionality according to its domain. However, it is possible that a particular functionality could belong to more than one domain.

According to the original FBS structure, we create a descending tree where the root is related to the IoT entity and the children define its functionalities. The final leaves of the FDBS tree will contain basic functionalities. For the FDBS, we create another parameter denoting which domain is considered for the proper functionality or sub-functionality (i.e., trust and privacy). As it is possible that two or more requirements could be the same and belong to different domains [7], thus, it is possible that a single functionality belongs to different domains. However, in this case we will not have two or more functionalities, but we will have a single functionality belonging to more than one domain.

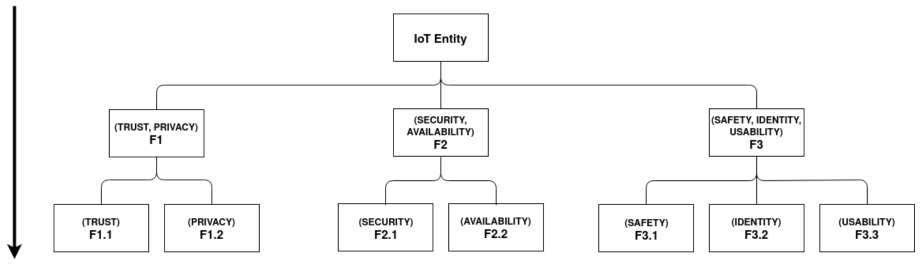


Figure 2: Functional Domain Breakdown Structure (FDBS)

Figure 2 shows an example of FDBS. There are three levels. The IoT entity is set at the top level. Then its main functionalities are set at the medium level, splitting them into sub-functionalities at the bottom level. As we can see, the domains are represented at the top of the box containing the functionalities. About this point, we want to remark for the reader that the IoT entity is composed of all the domains, but the functionalities can belong to a subset of domains.

However, in order to show how to implement our top-down approach in a use case scenario, we will present a specific example in Section 4.

3.2 Bottom-Up approach

For the bottom-up approach, the basic elements of a system are firstly analyzed and implemented. Then, the developer analyzes composed elements in order to proceed from a specific to a general view of the entity.

As we explained in Section 2.2, it is very important to take context into consideration during the SDLC of an IoT entity. Therefore, we will use this approach to model all the contexts belonging to the IoT entity under development. In fact, the IoT entities can participate in different contexts and some of them shall be separated from the others. Strongly related to the contexts, we also consider the domains (i.e., trust and security). However, there is the possibility that some contexts share functionalities and they can be considered together under a super-context. This super-context will include the single domains belonging to the contexts in the lower layer.

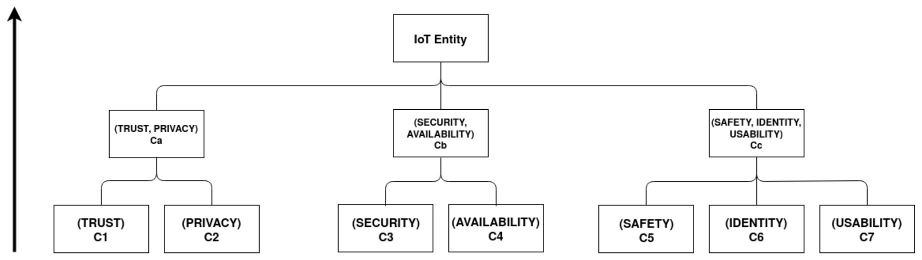


Figure 3: Bottom-Up Approach (Context)

The bottom-up approach is presented in Figure 3. It starts from the single contexts considering the ones at the bottom level and their domains. Then, the contexts having commonalities can be considered together under a super-context. In this general representation, we have three levels and the top level belongs to the IoT entity in its whole.

3.3 Trusted Block Development

In order to consider and develop software that is fundamental for any IoT entities, we propose a trusted block development.

This approach helps developers to delimit the software according to the contexts and functionalities highlighted in the bottom-up and top-down approaches.

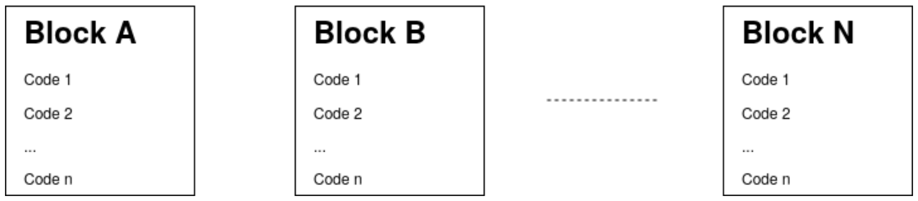


Figure 4: Trusted Block Development: each block is a *trust island*

This development style is fundamental to keep separated the different codes.

In Figure 4, we can see that the blocks are separated and they can contain several sub-blocks of code. They can be sequential or not, but it is important that they are separated. This is fundamental in order to preserve the separation among functionalities and contexts. This programming technique allows developers to create boundaries related to trust, creating "*trust islands*". In fact, if we consider variables existing only in a particular block, we can create trust operations according to the users or functionalities belonging only to that particular block. In fact, this design is helpful to keep the roles of a user separated according to a particular context. For example, a user can be considered and trusted in a block A, but it cannot be considered or not trusted in a block B. We will show an example of this part in Section 4.3.

3.4 Implementation Approaches

These approaches must follow a step-by-step methodology in order to be effectively implemented according to the K-Model proposed in Section 2.2. This methodology is presented in Figure 5.

The steps are the following:

1. The first step corresponds to the output of the previous phases of the K-Model (i.e., Need, Requirements and Model). In fact, all the tasks performed up to this step must be taken into consideration in order to develop the IoT entity. The needs specify the intended IoT entity, the requirements have been elicited according to them and the implemented models create useful guidelines to develop the IoT device in this central phase of the K-Model.

2. Then, in Sections 3.1 and 3.2 the proposed approaches are implemented in the second step. A context diagram will be very useful in the bottom-up approach. On the other hand, the other diagrams will be fundamentals during the implementation of the top-down approach.

3. Thirdly, as we specified in Section 3.3, according to both the approaches, we need to implement the trusted block development.

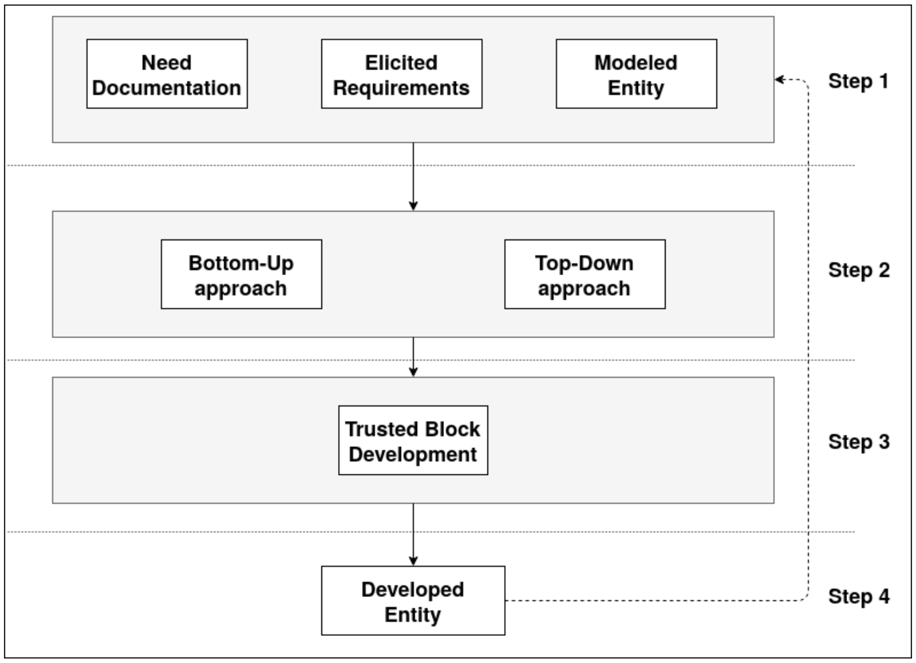


Figure 5: Step-by-step methodology

4. The fourth step of the methodology corresponds to the final developed entity that will be verified and validated in the following phases of the K-Model. Anyhow, it is possible to come back to step number one in the case some modifications are needed.

In the next section, we provide a use case scenario that realizes the implementation approaches presented in Section 3.

4 Use Case Scenario

As a use case scenario, we will expand the one presented in [7] implementing the approaches proposed in Section 3.

In this scenario, the IoT entity under development is a Smart Cake Machine needed to bake cakes and interfacing with other IoT entities: a smart fridge that is in the same smart home and smart super-markets belonging to the smart city environment. These connections are useful in order to check and order a particular ingredient, in the case it was needed for a recipe. Moreover, the IoT entity must allow trusted users to interact with it and deny the interaction for untrusted users.

In this section, we show how it is possible to apply the aforementioned approaches in order to develop the desired IoT entity. Anyhow, we will focus only on particular aspects also highlighted in [7]. In any case, it will be the developer the one in charge to choose which contexts and functionalities must be considered in order to develop the IoT entity according to the previous phases of the K-Model.

4.1 Top-Down approach

According to this approach, we need to follow a descending path starting from the consideration of the IoT entity as a whole and then going deep into the functionalities.

Analysing our previous work [7], we can state that some of the elicited requirements were related to the access control mechanisms. Furthermore, other requirements are necessary for the baking functionalities of the IoT entity.

Figure 6 presents the FDBS related to the Smart Cake Machine. Thus, we have the IoT entity on the top level. Then, in the second level, we have two general functionalities: access control and baking functionalities.

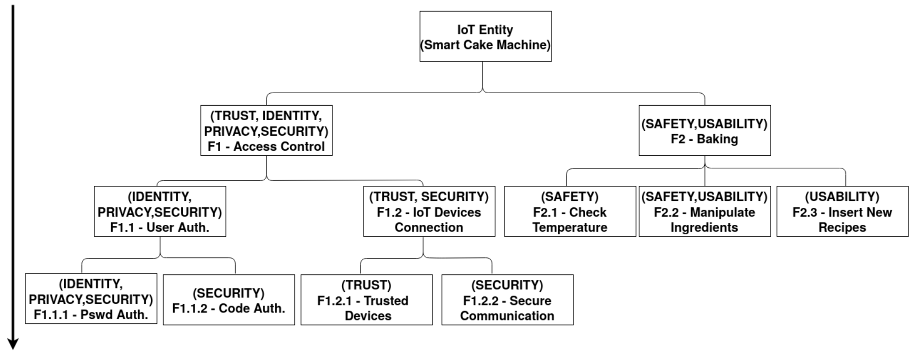

IoT Entity
(Smart Cake Machine)
(TRUST, IDENTITY, PRIVACY,SECURITY) F1 - Access Control
(SAFETY,USABILITY) F2 - Baking
(IDENTITY, PRIVACY,SECURITY) F1.1 - User Auth.
(TRUST, SECURITY) F1.2 - IoT Devices Connection
(SAFETY) F2.1 - Check Temperature
(SAFETY,USABILITY) F2.2 - Manipulate Ingredients
(USABILITY) F2.3 - Insert New Recipes
(IDENTITY, PRIVACY,SECURITY) F1.1.1 - Pswd Auth.
(SECURITY) F1.1.2 - Code Auth.
(TRUST) F1.2.1 - Trusted Devices
(SECURITY) F1.2.2 - Secure Communication

Figure 6: FDBS - Use Case: Smart Cake Machine

Considering access control, the related functionalities are divided into two fundamental parts: user authentication and IoT devices connections. In fact, the Smart Cake Machine can interact with trusted users and IoT devices. Concerning the users and according to our previous paper [7], we can consider two types of authentication: password and code authentication. The first one is related to the domains of identity, security and privacy. In fact, a password is related to a single user and it must be stored securely. Moreover, the data of the users must be collected and kept private. On the other hand, the code authentication can be considered without storing user information and it can be shared with other users whom the owner of the IoT device trusts. Moreover, this code must be securely provided.

About the connections between the Smart Cake Machine and the other devices, we need to consider the trusted devices guaranteeing that the communication among them is secure. So, the functionalities needed to create trust models for the devices will belong to the trust domain. The communication among these devices will belong to the security domain.

It is important to note that any functionalities belonging to this part of the "tree" is connected to the main functionality "Access Control". They are fundamentals to grant device access only to whom is trusted and can provide the credentials in order to interact with the device.

In Figure 6, we can see that the right part of the tree is related to the baking functionalities. In order to bake a cake, there are three fundamental functionalities. The first one is to check the temperature. It is a safety function for the correct preparation of the cake and for the safety of users and the device. The second one is related to the manipulation of the ingredients. This functionality is related to the safety and usability domain. In fact, safety is due to the health aspect. The usability and safety domains is related to the correct utilization of the machine manipulating the ingredients. Finally, the third functionality is related to the possibility of inserting new recipes. In this case, the usability domain is important because the interface must be user-friendly. As we can see, the domains are collected and separated according to the different functionalities.

4.2 Bottom-Up approach

This approach is useful to define the different contexts according to the Smart Cake Machine use case.

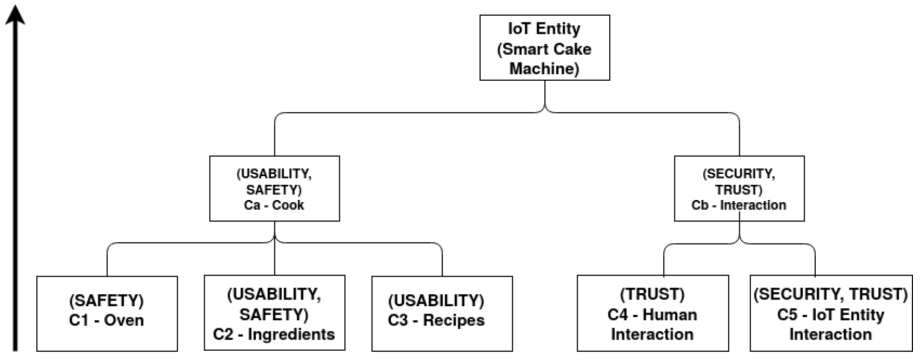

IoT Entity (Smart Cake Machine)
(USABILITY, SAFETY) Ca - Cook
(SECURITY, TRUST) Cb - Interaction
(SAFETY) C1 - Oven
(USABILITY, SAFETY) C2 - Ingredients
(USABILITY) C3 - Recipes
(TRUST) C4 - Human Interaction
(SECURITY, TRUST) C5 - IoT Entity Interaction

Figure 7: Bottom-Up Approach - Use Case: Smart Cake Machine

We can see the proposed contexts in Figure 7. We describe the contexts from right to left.

One context is related to the interaction among smart entities (i.e., smart super-market, smart fridge). Then, another context is related to the utilization of the device by human users. These two contexts can be considered together in a super-context related to the interactions. Trust and security are fundamental for these interactions and they are properly considered the main domains for these contexts. Then, we have a third context related to the recipes, another one about the ingredients and a fifth context related to the oven. These three contexts can be summarized into a super-context named cook context, where the considered domains are usability and safety. The two super-context are fundamentals for the trusted IoT Entity.

4.3 Trusted Block Development

After the utilization of bottom-up and top-down approaches, the developer must create the code according to them and to the previous phases of the K-Model. We present two different blocks in order to explain the different aspects of this approach and the differences among domains. The first block we present is *cook*, the second is related to *authentication*.

4.3.1 Block Cook

In this case, we start from the definitions of the contexts and functionalities creating separate code blocks according to them. We can see that contexts C1, C2 and C3 proposed in the bottom-up approach are also covered in the top-down approach with the functionalities F2.1, F2.2 and F2.3. In fact, it is possible to consider the same or similar aspects in both approaches.

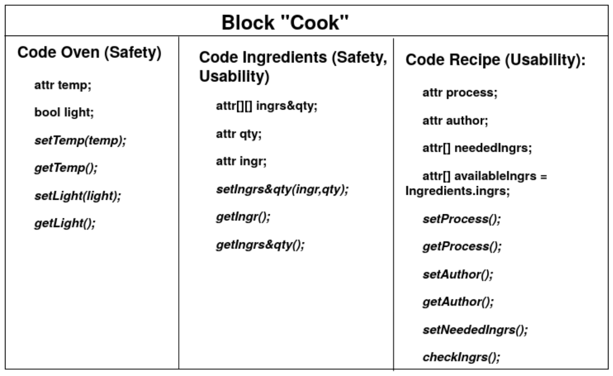
Thus, we will present the code block containing these highlighted aspects. For the first context and functionality related to the oven and the temperature, we need to set temperature attributes in order to fix predetermined levels related to different states during the cooking process. Then, for the ingredients, we need to consider them in order to be cataloged and inserted by the users. Finally, there are the recipes. We assume that they must be uploaded by the users or they can be memorized in the device by the vendors.

The domains are the same considered in the previous approaches (i.e., safety, usability) and they are declared at the start of the code blocks. We want to remind to the reader, that these domains are strongly connected to trust and their consideration allow developers to implement trust in the IoT entity.

In our example, we can see in Figure 8 that the block named *Cook* is composed of three parts: Oven, Ingredients and Recipe. We use the generic terminology attr (i.e., attribute) to consider characters, strings or numbers (i.e., integers, doubles, floats). Moreover, we use a Boolean variable and arrays of attributes. Then, we create methods useful to manipulate and check the attributes. For example, the attribute *setIngrs&qty(ingr,qty)* is fundamental to set which ingredient and its amount is considered. Another interesting method is *checkIngrs()*. It can be used to compare if the needed ingredients are available. We do not deeply specify the methods, we only declare them.

4.3.2 Block Authentication

Also in this case, we can start from the definitions of the contexts and functionalities by developing separate code blocks. In this case, we consider C4 (i.e.,


Block "Cook"

Code Oven (Safety)

```
attr temp;
bool light;
setTemp(temp);
getTemp();
setLight(light);
getLight();
```

Code Ingredients (Safety, Usability)

```
attr[][] ingrs&qty;
attr qty;
attr ingr;
setIngrs&qty(ingr,qty);
getIngr();
getIngrs&qty();
```

Code Recipe (Usability):

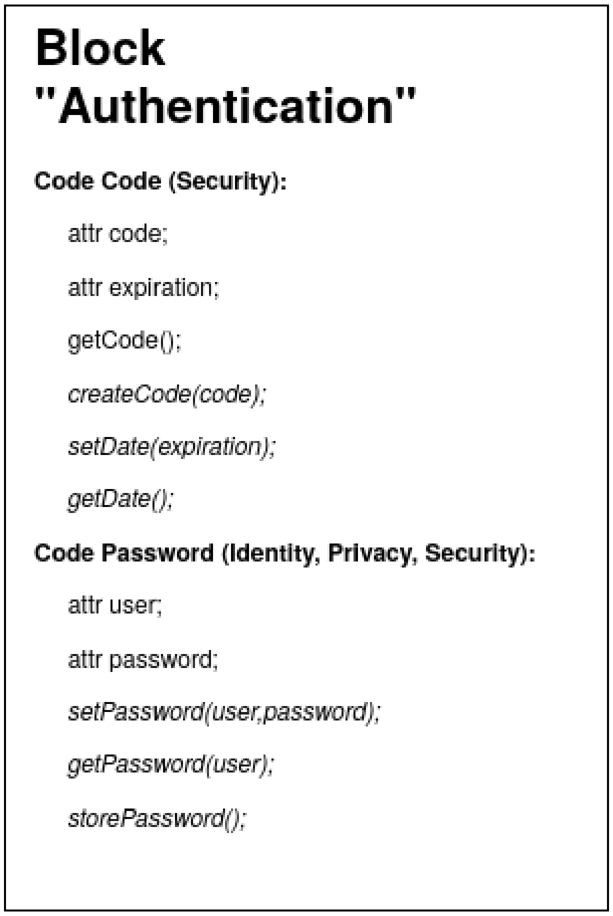
```
attr process;
attr author;
attr[] neededIngrs;
attr[] availableIngrs =
Ingredients.ingrs;
setProcess();
getProcess();
setAuthor();
getAuthor();
setNeededIngrs();
checkIngrs();
```

Figure 8: Trusted Block Development - Block Cook

Human Interaction) and the functionalities related to code and password authentication (i.e., F1.1, F1.1.1 and F1.1.2), In fact, in order to create and use a code or a password, the IoT entity must interact with the users. The example is narrow and we just want to give a hint on how to consider the steps in order to create separate code blocks.

Thus, for the first code named *Code*, we can see that it is connected to security and not to identity for example. This is because, in this use case, the code is unique and it can be used by different users in order to interact with the IoT device. On the other hand, we set different passwords according to the users in the second part of the block called *authentication*. In this case, each user has a password. We do not show exactly the procedure to create it, it depends on the user interface and the protocols that can be implemented. In this case, we want only to show how to distinguish between the different cases.

Therefore, we can see in Figure 9 that the block named *Authentication* is composed of two attributes: the code that can be generated and the expiration date. Then, according to the password code, we can see that we have to parameters: user and password. They are related and for another user it should not be possible to use the password of another. In this case, it is possible to separate the tasks related to a particular user, but this is another level of implementation that we do not discuss here. We wanted to present how the blocks must be implemented without going deeply with the specifications that are strictly connected to the use cases that will be considered.



```
Block
"Authentication"

Code Code (Security):
    attr code;
    attr expiration;
    getCode();
    createCode(code);
    setDate(expiration);
    getDate();
Code Password (Identity, Privacy, Security):
    attr user;
    attr password;
    setPassword(user,password);
    getPassword(user);
    storePassword();
```

Figure 9: Trusted Block Development - Block Authentication

5 Conclusion and Future Work

We have presented three approaches for the development of a trusted smart IoT entity during the core phase of the K-Model. These approaches allow developers to follow a schematic methodology in order to implement contexts and functionalities. Firstly, we have presented a top-down approach (FDBS) that is useful to specify domains and functionalities belonging to the IoT entity under development. Secondly, we have presented a bottom-up approach to implementing the domains and contexts belonging to the IoT entity under development. In this part, we need to start from the single contexts and aggregate them according to their scope under super-contexts. Finally, all the contexts will compound the IoT entity as its whole. As for the top-down approach, it is represented as a tree. Thirdly, we present a trusted block development considering the previous approaches. This development style is useful for the developers in order to consider all particularities of contexts and functionalities in single blocks of code according to contexts and functionalities. These three approaches are useful to develop the trusted IoT entity according to all the phases of the K-Model.

In future work, we will apply these approaches to a real use case scenario in order to validate them. Moreover, we will follow all the phases of the K-Model in order to provide the development phase with all the needed input and create the fundamental outputs for the following phases of the SDLC.

Acknowledgement

This work has been supported by the EU project H2020-MSCA-RISE-2017 under grant agreement No 777996 (Sealed-GRID) and the EU H2020-SU-ICT-03-2018 Project No. 830929 CyberSec4Europe (cybersec4europe.eu). Moreover, we thank Huawei Technologies 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] Carmely, T.: Using finite state machines to design software. EE Times (2009)

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

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

[4] Feamster, N., Rexford, J., Zegura, E.: The road to sdn: an intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review 44(2), 87–98 (2014)

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

[6] 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)

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

[8] 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)

[9] 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)

[10] Fraser, C.W., Henry, R.R.: Hard-coding bottom-up code generation tables to save time and space. Software: Practice and Experience 21(1), 1–12 (1991)

[11] Ganchev, I., Ji, Z., O'Droma, M.: A generic iot architecture for smart cities (2014)

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

[13] Jørgensen, M.: Top-down and bottom-up expert estimation of software development effort. Information and Software Technology 46(1), 3–16 (2004)

[14] Kozlov, D., Veijalainen, J., Ali, Y.: Security and privacy threats in iot architectures. In: BODYNETS. pp. 256–262 (2012)

[15] Moyano, F., Fernandez-Gago, C., Lopez, J.: A conceptual framework for trust models. In: 9th International Conference on Trust, Privacy and Security in Digital Business (TrustBus 2012. vol. 7449 of Lectures Notes in Computer Science, pp. 93–104. Springer Verlag (Sep 2012)

[16] Patel, P., Pathak, A., Teixeira, T., Issarny, V.: Towards application development for the internet of things. In: Proceedings of the 8th Middleware Doctoral Symposium. p. 5. ACM (2011)

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

[18] Reaidy, P.J., Gunasekaran, A., Spalanzani, A.: Bottom-up approach based on internet of things for order fulfillment in a collaborative warehousing environment. International Journal of Production Economics 159, 29–40 (2015)

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

[20] Valdivieso Caraguay, A.L., Benito Peral, A., Barona Lopez, L.I., Garcia Villalba, L.J.: Sdn: Evolution and opportunities in the development iot applications. International Journal of Distributed Sensor Networks 10(5), 735142 (2014)

[21] Van Kranenburg, R., Bassi, A.: Iot challenges. Communications in Mobile Computing 1(1), 9 (2012)

[22] Wagner, F., Schmuki, R., Wagner, T., Wolstenholme, P.: Modeling software with finite state machines: a practical approach. Auerbach Publications (2006)

[23] Xu, T., Wendt, J.B., Potkonjak, M.: Security of iot systems: Design challenges and opportunities. In: 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). pp. 417–423. IEEE (2014)

[24] Yan, Z., Zhang, P., Vasilakos, A.V.: A survey on trust management for internet of things. Journal of network and computer applications 42, 120–134 (2014)