Design optimization of IoT models: structured safety and security flaw identification

Julia Rauscher, Bernhard Bauer

Angaben zur Veröffentlichung / Publication details:

Rauscher, Julia, and Bernhard Bauer. 2020. "Design optimization of IoT models: structured safety and security flaw identification." In Business Modeling and Software Design: 10th International Symposium, BMSD 2020, Berlin, Germany, July 6-8, 2020; Proceedings, edited by Boris Shishkov, 84–102. Cham: Springer. https://doi.org/10.1007/978-3-030-52306-0_6.

Design Optimization of IoT Models: Structured Safety and Security Flaw Identification

Julia Rauscher (✉) and Bernhard Bauer

Software Methodologies of Distributed Systems, University of Augsburg, Germany
{julia.rauscher,bauer}@informatik.uni-augsburg.de

Abstract. More and more devices are being interconnected, thus extending the use of Internet of Things (IoT) systems. However, the larger the networks are the more vulnerable and inscrutable they become. This is a significant challenge especially when IoT is used in safety- and security-critical areas. In these areas, a flawless architecture must be guaranteed already in the design phase. Therefore, a structured possibility is needed to scan models completely for vulnerabilities as early as possible. We developed a pattern recognition framework (PRF) that enables the definition of design patterns and anti-patterns. These patterns are used for a holistic and automated identification of flaws in IoT models during design phase and enable a design optimization.

Keywords: Internet of Things, design optimization, pattern recognition, safety, security, by design, wellbeing

1 Introduction

In the age of digitalization there is an increasing number of devices which communicate and interact with each other. This has led to networks including more independent devices that can act and react in a uniquely identifiable and automated manner which are known as Internet of Things (IoT) systems. These systems have two major characteristics respectively challenges. First, they are increasing fast which creates complexity including hidden vulnerabilities. According to [1] by 2025 there will be 75.44 billion connected devices. Second, they aren't self-contained. Hence, they are connected to the internet that leads to possible cyber attacks and other threats. Therefore, IoT has to handle IoT-specific security challenges like data transmission in sensor networks as well as conventional issues like DOS attacks, eavesdropping or virus damages [2]. This set of challenges can arise and occur on plenty times and points. However, studies have shown that 50% of all flaws already emerge during the design phase [3]. Therefore, an approach to recognize vulnerabilities by design is urgently required [4]. Especially in safety- and security-critical areas these challenges are major concerns which require a reliable investigation of possible accidents or threats as early as possible. One example of the plenty application fields of IoT in safety- and security-critical systems is the deployment in the medical field. Not only in hospitals but also in private use as medical

smart homes IoT is used in the medical or wellbeing sector. However, this entails danger as shown by examples like [5] or [6]. Manipulated baby monitors or captured implantable cardiac devices represent highly critical elements which are difficult to alter afterwards and need observations during design phase.

Since most approaches focus on software level there is a lack of model-based IoT approaches. Additionally, the existing model-based approaches are either generic or cannot be automated. Therefore, to address the above mentioned issues, we have developed a pattern recognition process to present a structured approach to define and examine safety and security architectural patterns and anti-patterns. These are used to identify flaws during the design phase automatically. Thereby, flaws which can be prevented already before the run time are addressed. In addition, this approach enables knowledge conservation of safety and security design challenges and the review of large IoT systems. Since our process includes automated analysis parts, the complexity of IoT systems can be handled.

After introducing the challenges of safety and security in IoT systems, the remaining paper is structured as follows: To differentiate our approach from other concepts section 2 contains related work and background basics of allied fields. Afterwards, we will present the pattern recognition framework and its details in four steps. Section 4 applies an Ambient Assisted Living (AAL) use case to evaluate our diverse approach steps. An outlook on further work and the conclusion completes the paper.

2 BACKGROUND AND RELATED WORK

As described above, most approaches that identify flaws through patterns are conducted after the design phase. Accordingly, these approaches are software-based. Reference [7] offers a review over the attempts to create code patterns to identify bad software decisions. None of these attempts investigates the concept of patterns on architectural level. Another concept of using design patterns in data-intensive systems offers [8]. Though, they aim in detecting design patterns for reverse engineering purposes and not safety or security challenges. Approaches which include IoT and security by design are e.g. [9] and [10]. Reference [9] proposes the application of AADL to be able to model all security related information. Though, their framework doesn't include automated flaw identification possibilities. The work of [10] present the review of the usage of security design patterns in IoT systems. However, like [7] these patterns are made for software architectures. As is often the case, these two approaches do not consider safety.

Analyses on architectural level are often used in other application fields as IoT already. For instance, enterprise architecture management (EAM) applies architecture analyses for diverse goals and even for security analyses. An overview of the available analyses is offered by [11]. E.g. the approaches of [12] [13] [14] use defense graphs or extended influence diagrams to assess risks or other security concerns on design level. However, these architecture analysis approaches are only conducted if the vulnerabilities are already known. Therefore, an approach is required to identify these vulnerabilities to enable the application of the assessment.

This literature research has revealed that all concepts are already successfully used, but there is no approach which combines these concepts and applies them in the safety- and security-critical IoT area.

3 PATTERN RECOGNITION FRAMEWORK

Designing an IoT system flawless and without weak points is an almost unmanageable task. We developed an approach to support and simplify this task.

When planning the design or considering design changes two main issues occur. First, designing the details highly depends on expert experience. However, experts are not always available during design phase or afterwards if changes are required in the model. Therefore, a possibility to preserve the knowledge is highly significant. The second issue addresses the complex dependencies, interoperability and requirements of the IoT system components. Hence, a manually verification of all included elements, connections and features is not possible in a reasonable time.

Our approach of a pattern recognition framework (PRF) covers these both challenges. Thus, the PRF has multiple goals. To address the first challenge we created a selection of required information about flaws, risks, avoidable design decisions and possible impacts to offer a structured chance to enable a knowledge conservation. The experience of the experts can be saved for later observations. Therefore, misplaned design mistakes, which already happened before, can be prevented in the design phase during the design check. As the knowledge is conserved in textual readable form, every team member is able to follow it independent of programming language knowledge. To address the second challenge we translate the textual readable form into executable code by using a domain specific language (DSL). Thus, the flaw analysis can be conducted automatically and includes all components of the IoT system.

The knowledge conservation and flaw identification respectively recognition will be conducted through design patterns and anti-patterns. Patterns represent positive and desirable design choices that prevent vulnerabilities, e.g. a highly required authentication mechanism. To recognize possible flawed modelling decisions, model components that do not match these patterns are searched for. During this search, the model components are examined for conditions of the unambiguous pattern definitions. Matches are only displayed if they fulfill all pattern conditions to avoid false positive hits. By contrast, anti-patterns represent negative and avoidable design choices that cause vulnerabilities. The automated flaw identification looks for model components that match the anti-patterns. Our framework enables to define generally applicable patterns which are suitable to all IoT models. Though specifically designed patterns for individual IoT models are also possible and required. When we speak of pattern in the following, this includes anti-pattern as well. In addition, flaw identification and flaw recognition are used interchangeably as our concept is not related to the machine learning research field.

The application of the PRF is described below in four sections A-D. Section A explains the content of the PRF categories and attribute options, which are used to specify

the patterns and anti-patterns textually. Section B presents the pattern definition language. Finally, sections C and D show the transformation of the patterns into code and the automated, executable pattern recognition.

To illustrate the context, figure 1 provides an overview of the components and usage of our PRF during design time. Independently of the domain, the creation of a model requires a meta model. Therefore, we designed an IoT meta model which can be used to depict IoT networks. This meta model is among others able to depict physical entities, like sensors, actuators or tags, and their virtual counterparts. Additionally, including physical connections with their network specifications, protocol types and encoding mechanism. Furthermore, services with operations, right management and authentication requirements can be presented. In addition, components for business details, stakeholder and their users are contained. As described, the PRF (Section A) is applied to define pattern and anti-pattern in textual readable form. The DSL (Section B), which is based on the meta model, uses the fulfilled PRF parts, categories and values to configure the pattern language. A pattern database stores the created patterns and anti-patterns. Concurrently, a code generation process (Section C) produces executable code for pattern services, i.e. flaw identification services. When a flaw identification is conducted, the database and services are used to examine the IoT model (Section D).

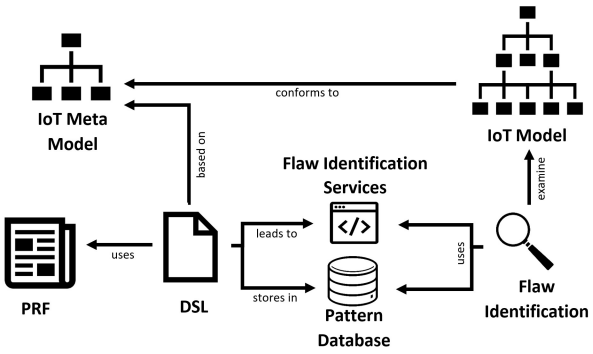

Fig. 1. Structured definition and identification of flaws by using PRF

To realize our approach we need different technologies. Therefore, we use the widely used concepts of EMF to model our IoT meta model. Our IoT models are depicted with the extension Eclipse Sirius [15], whereas the DSL and code generation are conducted with the related concepts Xtext and Xtend. [16] [17]

A. Pattern Definition

Following, the components of the PRF are explained in detail with TABLE 1 to TABLE 7. As the PRF covers different issues, we need a safety PRF type and a security PRF type. Both types consist of four pattern definition categories. However, the categories vary depending on the PRF type. The structure of the different PRF types consists of:

- a Generic Part (TABLE 1),
- a Safety OR Security Challenge Part (TABLE 2 OR TABLE 3),
- a Safety OR Security Assessment Information Part (TABLE 4 OR TABLE 5),
- a Pattern OR Anti-Pattern Implementation Part (TABLE 6 OR TABLE 7).

The first three parts are used for knowledge conservation and for later flaw and risk categorization or assessment. The fourth part will be used for the implementation details.

Attributes of the different parts are either of free text style that don't underlay bounds or of enumeration style. Enumerations are predefined sets of possible values.

Every PRF type starts by using a *Generic Part*, which is presented in TABLE 1 to specify the conditions that are independent of safety or security specific characteristics. To be able to identify the defined patterns in the pattern database an ID and name is required. Apart from this a supercategory for the protected element is included. This supercategory is used for pattern categorization within the pattern database and defines the element type and category, as well as the user group type. The element to protect categories are extracted of the IoT-A project. [18] These categories attempt to cover all aspects of an interactive IoT system and were elaborated of a special IoT security architecture approach. Therefore, users, different kind of devices, software and hardware aspects are covered. Since the kind of hard- and software is decisive of the needed actions, the type of physical connections and services is specified. These attributes are dependent on the used meta model. To conduct analyses on diverse model levels, e.g. layered protection analysis, the affected layer of the pattern must be set. Most IoT architecture approaches use 3-4 layers. Since safety and security issues need a more specific categorization, we chose a more detailed approach. For this purpose we used the layered architecture approach of [19] which consists of eight layers. We extended the approach by a user layer. To determine the responsible stakeholder the location of vulnerability is specified. This attribute helps to divide the architecture decisions. The last attribute views the disruption tolerance to categorize the sensitivity of the affected element. While tolerant and temporary tolerant elements perhaps can endure attacks, zero tolerant elements are highly critical.

Table 1. PRF Generic Part

Pattern Recognition Framework	
Generic	
ID	*free text*
Name	*free text*
Component	Supercategory: Element to protect
Element Type	*free text*
User Group	*free text*
Element category	Choice: Physical Person, Communication Channel, Leaf of Devices, Intermediate Devices, Backend, Infrastructure, Service, Facilities
HW	*free PhysicalConnection type*
SW	*free Service type*
Architecture layer	Choice: One layer of IoT Layered Architecture
Location	Choice: Local or Cloud
Disruption tolerance	Choice: Tolerant, Temporary tolerant, Zero tolerant

Next definition step is the *Safety* or *Security Challenge Part*. TABLE 2 defines the specific characteristics security challenges bring along to vulnerabilities. The intent and risk represent the aim and risk of loss of possible attacks. To classify the type of attack the STRIDE categorization is used. [20] The letters represent: Spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege. For instance, a distinction is made between spoofing and tampering that are indications how to prevent a weak point. As the goal of attack needs classification as well, an enumeration for attack goals is provided, which is inspired by [21]. It is distinguished whether

an attack aims at perhaps less critical information disclosure or in destroying/manipulating a whole network or functions. A capture attack tries to get control or access of an IoT device or critical data. This attack does not necessarily have direct impacts. However, they enable other attacks like DDDD (Disrupt-Degrade-Deny-Destroy). DDDD aims at affecting a system and disabling important functions. These goals bring along manipulation and attacks on diverse points. All these categories are used for assessment and database usage, too.

Table 2. PRF Security Challenge Part

Security Pattern Recognition Framework	
Security Challenge	
Intent	*free text*
Risk	*free text*
Classification	Choice: STRIDE: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege
Attack Goal	Choice: Capture, Disrupt-Degrade-Deny-Destroy, Manipulation, Information Disruption, Host Attack, Network Attack

As described above, every security PRF category has a corresponding safety category which includes safety specific characteristics (TABLE 3). Within this part the fault and fault class are determined. The fault attribute describes the possible origin of failure. Whereas the fault classes define the possible type of failure. These classes range from hardware causes like attrition to software or logical causes like interface issues or miscalculation. In addition, the hazard describes the consequences of possible faults which can also be classified. For instance, the types distinguish between simple failures, complete outages, single service losses or a system corruption. External problems can be mentioned as well, however these issues are difficult to prevent. Though, countermeasures or security measures can be taken into consideration. These both enumeration categories are extracted of [22].

Table 3. PRF Safety Challenge Part

Safety Pattern Recognition Framework	
Safety Challenge	
Fault	*free text*
Hazard	*free text*
Classification	Choice: Failure, Outage, External Problem, Loss of Service, System Corruption
Fault Class	Choice: Attrition, Energy, Calculation, Change Impact, Configuration Management, Data, Interface, Logic, Omission, Timing, Initialization

After specifying the *Generic* and *Safety/Security Challenge Parts* the details for further assessments of possible risks or hazards can be determined.

The assessment information of security flaws is shown in TABLE 4. Since assessment information can vary by domain and review reason, the details can be registered free without restrictions. Privacy is often related to security and is taken into consideration during assessment as well. Therefore, personally identifiable information (PII) can be mentioned to ensure attention to this aspect. The *Assessment Information Part* has a supercategory as well, to describe the possible direct impacts and their estimated consequences. These values are based on probability values and are subject to estimations or previous experiences. As direct impacts can vary widely depending on the system an exemplary choice range is given in the framework. The most important aspects are security typical issues like availability, functionality, manipulation of nodes and disclosure of data or structure aspects that allow further attacks. However, the range can be just

extended. For every direct impact an estimated probability value can be assigned. E.g. 'Availability 30% down'. These values refer to the defined elements to protect of the *Generic Part* and are used for further impact analyses. Next to direct impacts, also indirect impacts exist. These kinds of impacts are more difficult to estimate and often not obvious. Therefore, the type of indirect impacts can be chosen freely and offer hints for system architects for further design decisions. Related to the impacts is the seriousness of attacks. We extracted our categories from [23] as these are typically used distinctions. An attack can be catastrophic and cause complete system failure, as a critical attack generate issues in important parts, which e.g. are connected with confidential data. Marginal or negligible attacks create threats with moderate or conquerable impact. Attacks with no severity are more or less insignificant as they are not attack critical nodes or data. As last attribute of the *Security Assessment Information Part,* the security requirements are chosen. These requirements include the typical CIA (confidentiality, integrity and availability) security aspects. Since these aspects do not cover all possible IoT issues, the enumeration was extended. For instance, authentication and non-repudiation was added. Authentication is one of the most important and vulnerable aspects of IoT as this process enables the interaction with devices and the access to important data. Whereas non-repudiation is required to ensure the traceability of actions, measurements and node extensions.

Table 4. PRF Security Assessment Information Part

Security Pattern Recognition Framework	
Assessment Information	
Assessment Information	*free text*
PII	*free text*
Probable Direct Impacts	Supercategory: Estimated Consequences
Impacts	Choice: Disclosure, Manipulation, Availability, Functionality
Values	*free probability values*
Indirect Impacts	*free text*
Severity	Choice: Catastrophic, Critical, Marginal, Negligible, Non
Security Requirements	Choice: Confidentiality, Integrity, Availability, Manipulation Resistance, Privacy, Authentication, Non-Repudiation

The counterpart of the security assessment displays the *Safety Assessment Information Part*, which is shown in TABLE 5. Some aspects are equal to the security assessment. However, these aspects are not included in the *Generic Part* as the values of equal attributes still depend on safety or security specific issues. First attribute of the safety assessment also defines the free information details depending on the domain and assessment reason. Since there are several possible reasons, also multiple information can be set. Likewise, a supercategory for elements to protect is included for estimating the consequences of direct impacts. The enumeration of impacts can be extended as well. These impact categories overlap with security impacts since safety and security are related and cannot be separated completely. Impacts on functionality and availability can cause life-threatening behavior, whereas reliability represent the correct outcome of safety-critical actions. Like the security assessment, safety will be assessed with estimated probability values, too. Since the knowledge on this point also comes largely from experts, knowledge conservation is important. A more critical aspect of safety assessment are indirect impacts as these can cause new safety-critical aspects that can harm human beings. Equally, this challenge is often dependent on experience as well. Safety patterns require other categories for assessment like security since they cover the well-being of users. The severity divides between deathly, serious, e.g. internal injuries, and non-serious accidents. [24] Depending on this division, the patterns are ranked in a

higher priority within design decisions. An approach to rate the likelihood of occurrence is offered by [23]. Which frequency corresponds to which category depends on the internal guidelines. However, the rough categorization are frequent and probable occurrences for highly possible accidents, whereas less possible accidents are divided in occasionally and improbable. Improbable accidents are defined, because the likelihood of occurrence can change in the future. Since safety brings along its own needs, we specify safety requirements. For instance, typical requirements are the ability of recovery of devices or functions, redundancy of sensitive elements and data/device integrity to ensure right calculations/services.

Table 5. PRF Safety Assessment Information Part

Safety Pattern Recognition Framework	
Assessment Information	
Assessment Information	*free text*
Probable Direct Impacts	Supercategory: Estimated Consequences
Impacts	Choice: Functionality, Availability, Reliability
Values	*free probability values*
Indirect Impacts	*free text*
Severity	Choice: Death, Serious Injury, Non serious Injury, No Injury
Likelihood of occurrence	Choice: Frequent, Probable, Occasional, Improbable
Safety Requirements	Choice: Recovery, Redundancy, Failure Resistance, Availability, Data Integrity, Device Integrity

Until this point, the pattern definitions create the basis for the knowledge conservation. Therefore, following parts are responsible for the implementation and automated flaw detection. Additionally, the implementation parts are not safety or security specific. However, they are pattern or anti-pattern intrinsic. Thus, the implementation part of patterns defines positive design decisions that prevent accidents or threats. By contrast, the implementation part of anti-patterns specifies negative design decisions, which cause exploitable or erroneous vulnerabilities. TABLE 6 presents the *Pattern Implementation Part* that is used to design a desirable architecture. For documentation reasons a textual solution initiates the implementation details. On this occasion a short description of the concrete element, relation and attribute types should be given. This description is displayed after the flaw identification to explain the discovery and to prevent misconceptions. A supercategory includes the specification of required combinations of nodes, attributes and their relations. These combinations represent the elements to protect and their risks. Before specifying the concrete elements, an algorithm is set. How the elements are dealt with and are used for the identification depends on these algorithms. For instance, nodes' attributes can need comparison with other conditions or a simple summation of nodes is required. Section C is reliant on these algorithm types to conduct the code generation appropriate. Following the concrete node, relation and attribute types are defined. The available types depend on the used IoT meta model. All depictable elements and characteristics must be selectable to check the whole IoT model and to define patterns for all aspects. All nodes connected to the pattern must be selected. Accordingly, the next step is to select the relations that connect these nodes. Finally, the attributes which the nodes must fulfil to be affected by the pattern are selected. To highlight the flaw, the last point of the PRF is a textual documentation of the exact flawed feature.

Table 6. PRF Pattern Implementation Part

Safety/Security Pattern Recognition Framework	
Implementation	
Solution	*free text*
Implementation	Supercategory: Required "Node X Attribute X Relation" Combination
Pattern Algorithm	*free text* (examples: Summation of nodes)
Node2Node	Choice: Available Node types
Node2Relation	Choice: Available Relation types
Node2Attribute	Choice: Available Attribute types
Flaw	*free text*

Our last part of the PRF is shown in TABLE 7 and is the counterpart of the pattern implementation. The *Anti-Pattern Implementation Part* is similar to the pattern definition. However, as an avoidable design will be described not a solution will be documented. Thus, a security-specific anti-pattern explains a vulnerable design, while a safety-specific anti-pattern documents a hazardous design. Both variants are used for documentation and explanation after the flaw recognition in the IoT model. Furthermore, there is also a supercategory for the required combination of node, attribute and relation types as stated above. The process starts in the same way as mentioned before with a pattern algorithm definition. As before, the algorithm is responsible for further code generation activities and the general handling with the defined elements. Following the contained meta model types are chosen including the related connections and features. Once again, the flaw characteristic description of the element to protect finishes the PRF anti-pattern section.

Table 7. PRF Anti-Pattern Implementation Part

Safety/Security Pattern Recognition Framework	
Implementation	
Vulnerable/ Hazardous Design	*free text*
Implementation	Supercategory: Required "Node X Attribute X Relation" Combination
Pattern Algorithm	*free text* (examples: Summation of nodes)
Node2Node	Choice: Available Node types
Node2Relation	Choice: Available Relation types
Node2Attribute	Choice: Available Attribute types
Flaw	*free text*

The presented categories and their enumerations are adaptable and extensible if the used domain requires other topics or values to define or review the model.

B. Pattern Specific Language

Through section A the PRF contained categories and values have been clarified. At this stage, however, the defined patterns and anti-patterns cannot yet be used for automatic flaw detection. Therefore, the patterns must be configured using a DSL. As the domain of our approach is pattern detection, we use Xtext to create a pattern definition language. The patterns are stored in the database in this form. Following, we describe the structure and rules of this DSL.

Our language contains all parts described in section A and is also structurally oriented towards them. The presented category choices are realized with Xtext respectively Ecore enumerations. These can be extended at this point as well, if the analyzed IoT model requires changes. To enable the full configuration of patterns and complete review of IoT systems, the DSL also contains the IoT meta model elements including

all node types, attribute types, relation types and type enumerations. If other safety or security specific features are needed, these can be added on our pattern definition language, too. Since the language cannot be presented fully, by means of code examples an insight should be given.

Figure 2 displays a code extraction of a Xtext pattern definition including the initial feature **patterntype** within the initial start rule 'PatternDefinitionFramework' (Line 5-7). The Xtext rule 'PatternType' (Line 9-11) delegates either to the rule 'SecurityPattern', 'SafetyPattern', 'SecurityAntiPattern' or 'SafetyAntiPattern'. We consider the rule 'SecurityPattern' exemplarily that starts with a keyword. The rule contains four features each adding one of the PRF parts which are available to choose from. Each of the other rules that are delegated from 'PatternType' are structured in this way, however, with its customized choice range.

```
5 //Choice of pattern type
6 PatternDefinitionFramework:
7     patterntype=PatternType;
8
9 PatternType:
10    SecurityPattern|SafetyPattern|SecurityAntiPattern|SafetyAntiPattern
11;
12
13 //SecurityPattern definition
14 SecurityPattern:
15    'SecurityPattern' '{'
16    generic+=GenericPart
17    challenge+=SecurityChallengePart
18    assessment+=AssessmentInformationSecurityPart
19    implementation+=ImplementationPatternPart
20    '}'
21;
```

Fig. 2. Xtext code to define the PRF as DSL

Afterwards the rules for the generic part, challenge parts, assessment information parts and implementation parts are defined. To offer an insight in one of these parts figure 3 views the rule 'AssessmentInformationSecurityPart' to configure the assessment details. The rule contains multiple features and keywords (introduced in TABLE 4). These features are labelled as identifier, String value or a kind of enumeration. The feature **securityRequirements** can add an arbitrary number of values, as several requirements can be needed. The feature **directImpacts** is a special feature as it is a type of the rule 'ProbableDirectImpacts' (Line 109-116). This rule configures impacts which can be chosen from security or safety impact enumerations. As impacts don't necessarily appear these values are optional. The concrete values are described with the rule 'Change' and an Integer probability. This rule enables to depict the direction (up or down) of changes.

```
97 //Assessment Information Security Part
98 AssessmentInformationSecurityPart:
99     'AssessmentInformationSecurityPart' '{'
100    'assessmentFeature' assessment=ID
101    'pii' pii=STRING
102    directImpacts=ProbableDirectImpacts
103    'indirectImpacts' impact=STRING
104    'severity' ':'  severity=SecuritySeverity
105    'securityRequirements' '{' (requirements+=SecurityRequirements(',')?)* '}'
106    '}'
107;
108
109 ProbableDirectImpacts:
110    'ProbableDirectImpacts' '{'
111       ('Impacts' ':'  (impactSE+=SecurityImpact)? (',')? (impactSA+=SafetyImpact)?
112       'values' ':'
113       change+=Change
114       value+=INT '%')*
115    '}'
116;
```

Fig. 3. Xtext code to define the Security Assessment Information Part

As last part the implementation parts must be specified with Xtext rules. Figure 4 displays exemplarily a rule for safety anti-patterns. The hazardous design can be described with a String value, whereas the implementation elements themselves depend on the 'Implementation' rule. Every **rule** feature requires an identifier for identification reasons, as well as the **type**. The conditions and correlations of pattern elements are defined with keywords like 'if', 'then' or 'equals'. A **flaw** is determined with specific nodes that were defined with the 'Implementation' rule.

```
//Implementation Anti-Pattern Safety Part
ImplementationSafetyAntiPatternPart:
    'ImplementationSafetyAntiPatternPart' '{'
    'hazardousDesign' design=STRING
    implementation=Implementation
    'flaw' (('and')? flaw+=[Node|STRING] ('=' flawValue+=STRING)?)*
    '}'
;

Implementation:
    'PatternRule' rule=ID '{'
    'algorithmType' type=ID
    ('condition' conditionNum+=ID)*
    (('if')? ('and')? ('then')? ('then' 'not')?
        ('equals')? ('or')? node+=Node
    )*
    '}'
;
```

Fig. 4. Xtext code example of the Anti-Pattern Implementation Part

Beside the described or mentioned rules, enumerations, PRF specific rules and meta model rules are components of the pattern definition language.

C. Pattern Service Generation

After concluding the first two manually conducted steps the theoretical pattern definition is completed. To enable the automated flaw identification, applicable pattern services are required. To transform the structured, defined patterns and anti-patterns into executable code an automated code generation will be conducted. Part four (TABLE 6 and TABLE 7) contains the required implementation details to enable the recognition of patterns or anti-patterns in a model. The transferred details of a DSL (Section B) are translated automatically into the needed services through the functions of Xtend. As described before in section A, the code generation acts depending on the chosen pattern algorithm. Once, a pattern or anti-pattern was defined and saved in the pattern database, an automated code generation is conducted and saved as well. Therefore, every specified and saved pattern can be used for design optimization immediately. This code generation process is provided for all kind of pattern.

D. Pattern Identification

The last step of our PRF represents the final automated identification of vulnerable or hazardous design flaws in IoT models. The created services (Step C) will be used to analyze and optimize the design automated. The results of the pattern recognition process depend on the kind of defined pattern. In case of identification of an anti-pattern all elements and relations, which match with the definition, are highlighted as they are follow a negative design decision. However, if a pattern identification is conducted to recognize flaws, only the elements that are not matching the definition are highlighted, as the definition represent desirable design. Every IoT Model that complies with the developed IoT meta model can use the content of the pattern database to detect flaws.

4 Evaluation

After explaining the details of the PRF we demonstrate the application to evaluate our approach steps. As described before the usage of IoT in medical cases implicate safety- and security-critical aspects. Therefore, we designed a smart home in manner of an AAL use case for elderly. To create an approved use case for evaluation we take into consideration the approaches of [25], [26] and [27]. As the devices in our AAL smart home aiming in prevention, it is a wellbeing use case. We define medical devices as wellbeing devices if their main purpose is monitoring, tracking or detecting, i.e. prevention, and do not have direct impact on the body. Figure 5 shows the rough structure and contained devices respectively infrastructure. In our smart home an elderly resident is monitored through multiple devices with different locations and goals throughout the house. For instance, the defibrillator is a fully implanted device to monitor the heart, whereas the insulin pump is a kind of wearable to measure the current insulin level. The other devices are located in diverse places, as the fall detector is positioned in every room and the mobile phone and pillbox vary. To collect and process the data two kind of cloud respectively gateway are included. In addition, the smart home is connected with diverse stakeholder with different goals and rights, like ambulance for critical situations or relatives who may inform about the resident. The network details of our use case were presented in [19].

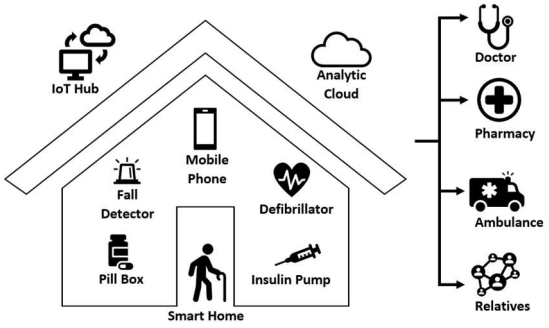

Fig. 5. AAL use case scenario used for evaluation

Our detailed AAL use case architecture and dependencies are modelled based on the developed meta model for IoT systems. Figure 8 shows an excerpt of the IoT system of our AAL use case which conforms to our meta model. In this excerpt some of the included devices with their physical connections, services and users are depicted.

To evaluate the sections A-D of chapter III, we will present two challenges. For these challenges we will define a pattern respectively an anti-pattern with our PRF to validate the structured way of knowledge conservation. Afterwards, the patterns will be transferred into DSL to enable the code generation for the flaw identification services. To present the added value of our framework, we show the results of the automated flaw identification process applied on our AAL use case model. Following, two safety respectively security challenge examples are considered in detail:

- Example 1 (Security): IoT devices with limited space and energy are easy entry points for cyber attacks as the encryption is missed or neglected
- Example 2 (Safety): Authentication can be a safety issue if life-saving functions are blocked by these methods

For example 1 a pattern will be defined and example 2 is covered through an anti-pattern definition. For both example challenges the PRF is used to define the pattern respectively anti-pattern. TABLE 8 shows the pattern definition of example 1. The pattern receives an ID and the name *'Lightweight devices without encrypted communication'.* The affected devices in our use case are *defibrillators* which are a kind of *well-being IoT devices* and a *leaf of devices*. These are only used for monitoring reasons as they are wellbeing devices and not medical devices in our use case. To communicate with other devices or gateways a *Bluetooth connection* is applied. As described before a *monitoring service* is required which can be affected. Since IoT devices are the basis of every IoT system they are on in the bottom of the layered architecture (*Thing Layer*). However, the location of security attack can be either local or in a cloud. As the defibrillator is only a wellbeing device it is *temporary tolerant* towards attacks. The security challenge aims in *capturing and changing private data on the communication way* that can cause the *theft of PII* and *manipulated analyses results,* as the attack can be classified as *tampering* and the attacker's goals are *capture* and *manipulation.* To assess the impacts of this kind of attack the *integrity* must be taken into consideration to evaluate the correctness of the *measured cardiac conditions.* Since the alteration of data can have impacts on the *functionality* most likely, an estimated change of *70% down* is expected. In addition, *other health recommendations are affected* since the holistic health status is impacted as well. Altered cardiac measurements are of a *critical* nature as these data are used for long-term analytics, e.g. atrial fibrillation monitoring. Security requirements which are affected through a possible attack are *confidentiality, integrity, manipulation resistance* and *privacy.*

Table 8. Pattern Definition of Security Example 1

Pattern Recognition Framework	
Generic	
ID	876543
Name	Lightweight devices without encrypted communication
Component	Supercategory: Element to protect
Element Type	WellbeingIoTDevice (Defibrillators (monitor cardiac conditions))
User Group	Patients
Element category	Choice: Leaf of Devices
HW	Bluetooth Connection
SW	Monitoring Service
Architecture layer	Choice: Thing Layer
Location	Choice: Local, Cloud
Disruption tolerance	Choice: Temporary tolerant
Security Challenge	
Intent	Capture private information, Change of data on comm. way
Risk	Theft of PII, Manipulate data and change analysis results
Classification	Choice: Tampering
Attack Goal	Choice: Capture, Manipulation
Assessment Information	
Assessment Information	Integrity
PII	Cardiac Condition, Measurement values
Probable Direct Impacts	Supercategory: Estimated Consequences
Impacts	Choice: Functionality
Values	70% down
Indirect Impacts	Impact on other health recommendations
Severity	Choice: Critical
Security Requirements	Choice: Confidentiality, Integrity, Manipulation Resistance, Privacy

After we conserved the knowledge of the security attack we define the implementation details to recognize the vulnerability (TABLE 9). The solution for our example 1 is based on the recommendation of [25]. They suggest prohibiting devices with weak or no encryption algorithms the direct communication to a cloud. These devices should use a field gateway as an intermediary. The field gateway will encrypt the data and deliver them to the analytic cloud. Therefore, the pattern recognition uses an *attribute validation* algorithm to check the types of used gateway. The included nodes are of node types *WellbeingIoTDevice, PhysicalConnection, Encryption* and *Gateway.* Node2Relation specifies the relations between the chosen node types. In addition, the nodes must have corresponding attributes to be included in the pattern. *WellbeingIoTDevices* have to be of the *deviceType: Defibrillator,* whereas their *PhysicalConnections* must use a *Bluetooth protocol*. If the *encryptionType* is *undefined,* the *Gateways* have to be of the type *FieldGateway.* However, if the *gatewayType* is not of type *FieldGateway* a vulnerable design is present.

Table 9. Pattern Definition of Security Example 1

Pattern Recognition Framework	
Implementation	
Solution	Device with weak encryption algorithm has to use a field gateway to connect with a cloud gateway
Implementation	Supercategory: Required "Node X Attribute X Relation" Combination
Pattern Algorithm	Attribute Validation
Node2Node	Choice: WellbeingIoTDevice, PhysicalConnection, Encryption, Gateway
Node2Relation	Choice: physicalconnection, encryption, peereddevice
Node2Attribute	Choice: deviceType: Defibrillator, protocol: Bluetooth, encryptionType: undefined, gatewayType: FieldGateway
Flaw	Flaw: gatewayType= not FieldGateway

After a pattern was defined to prevent security challenges like example 1, the pattern will be transformed into the DSL. Following, we present a Xtext extract of an anti-pattern for example 2 (Figure 6) and afterwards another snippet of the transformed example 1 (Figure 7).

As mentioned above we define an anti-pattern for our safety example 2. This negative design decision plans multi-way authentication methods before enabling ambulance calls. Therefore, we use the Xtext rule 'SafetyAntiPattern'. Figure 6 shows the transformed PRF for the *Generic Part, Safety Challenge Part* and *Safety Assessment Information Part.* For the generic information the **ID**, **name**, **component** details and the other categories were transformed. E.g. as an ambulance call corresponds to an acting service of a mobile phone, a 3G connection is required and the affected layer is the *SensingActingLayer.* The DSL also transformed the safety challenge information like possible hazard (*Help cannot be contacted in time*) and classification of this (*Loss of Service)*. As last part of the knowledge conservation of our safety anti-pattern the assessment information are transformed. As a service authentication implies *confidentiality* changes this feature will be used for assessment. Depending on the used authentication method indirect impacted misuse of ambulance calls can happen. The implementation part of the anti-pattern specifies the hazardous design of 2-way authentications connected with ambulance calls.

```
SafetyAntiPattern{
    GenericPart{
        id SAAP927392
        name UrgentSituationAuthentication
        Component{
            elementType{ Service }
            userGroup{ undefined }
            elementCategory{ Service,Facilities }
        }
        hardware "3G Connection"
        software "Acting Decision Service"
        layer: SensingActingLayer
        location{ Cloud }
        disruptionTolerance:Temporary tolerant
    }
    SafetyChallengePart{
        fault "Complicated authentication method for ambulance call"
        hazard "Help cannot be contacted fast enough"
        classification{ LossofService }
        faultClass{ ConfigurationManagement,Initialization }
    }
    AssessmentInformationSafetyPart{
        assessmentFeature ConfidentialityAssessment
        ProbableDirectImpacts{ Impacts:Confidentiality values: up 10% }
        indirectImpacts "Misuse of ambulance"
        severity: SeriousInjury
        likelihood: Probable
        safetyRequirements{ Availability }
    }
```

Fig 6. Defined Safety Anti-Pattern Xtext code example 2

The same transformation has to be conducted for example 1. Figure 7 displays the implementation part of our security pattern, whereas lines 52-66 show the conditional pattern rule. As described above **if** a *WellbeingIoTDevice* with a *Physical Bluetooth Connection* has no defined *Encryption* **then** the *Gateway* has to be a *FieldGateway*, otherwise a flaw exists.

```
47  ImplementationPatternPart{
48      solution "Device with weak encryption algorithm has to use a field gateway
49          to connect with a cloud gateway"
50      PatternRule SEP8765432_1{
51          algorithmType AttributeValidation
52          if
53          WellbeingIoTDevice WellbeingIoTDeviceNameSEP8765432{
54              deviceType Defibrillator
55              physicalconnection ( "PhysicalConnectionNameSEP8765432" )
56              gateway ( "GatewayNameSEP8765432" )}
57          and
58          PhysicalConnection PhysicalConnectionNameSEP8765432{
59              protocol Bluetooth
60              encryption{
61                  AsymmetricEncryption AsymmetricEncryptionNameSEP8765432{
62                      type undefined }}}
63          then
64          Gateway GatewayNameSEP8765432{
65              gatewayType FieldGateway
66              peereddevice( "WellbeingIoTDeviceNameSEP8765432")}}
67      flaw
68          "GatewayNameSEP8765432" = "not FieldGateway"
69          and
70          "PhysicalConnectionNameSEP8765432.AsymmetricEncryptionNameSEP8765432" = "undefined"}
```

Fig. 7. Defined Security Pattern Implementation Xtext code example 1

Afterwards, the DSLs of example 1 and 2 are translated into pattern services, i.e. flaw identification services, with Xtend to enable their automated usage. In addition, they are stored in the pattern database.

Since the pattern definition process is finished, we are able to evaluate our last concept step: The flaw identification and model optimization. Figure 8 shows our modelled and analyzed AAL use case with all included physical entities, physical connections, services, clouds, stakeholder, users and their relations among themselves. The elements

are colorized depending on their layer type. To optimize this model we apply our automated flaw identification process through the execution of the generated services with Eclipse Sirius Services which highlight the vulnerable elements and their relations in red color: The flaw identification services found the elements and relations ‘PhyCo_Hub2Defi’, ‘SHDefibrillator’ and ‘SHIoTHub’ as this design is not matching our positive design pattern for IoT devices without adequate encryption types and the used gateway type, as ‘SHIoTHub’ is not a field gateway. As well, identified weaknesses are e.g. ‘PhyCo_AmbulanceCall’, ‘Ambulance Call_Service’ and ‘SHSmartPhone’. As the service ‘Ambulance Call_Service’ requires a multi-way authentication and endangers residents, this design matches our negative anti-pattern definition of blocked ambulance calls.

After applying the pattern recognition the avoidable design decisions can be optimized. The evaluation of our concept was conducted exemplarily on a medical use case. However, the application of the framework is able be deployed in all domains of IoT to optimize the design and identify design smells.

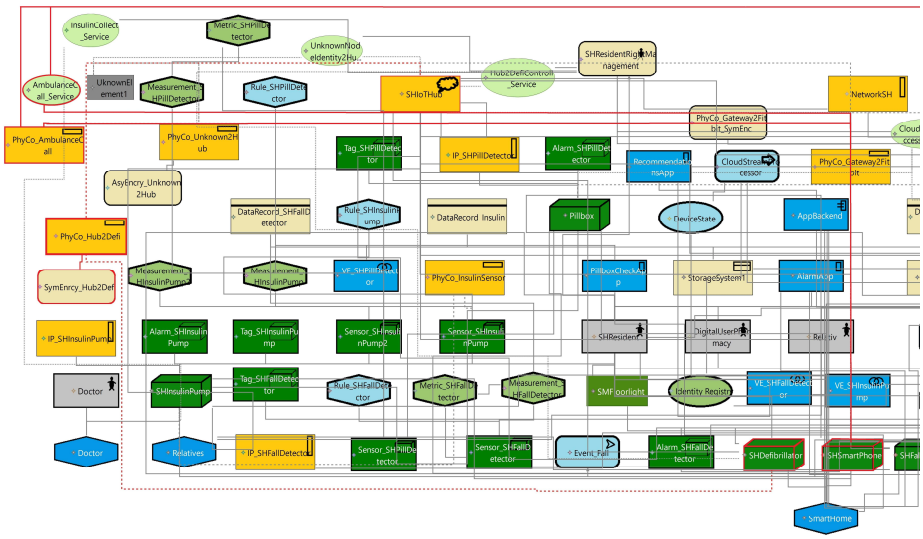

Fig. 8. Flaw identification within the AAL use case model

5 Conclusion

In the previous sections, we presented our approach to identify design flaws. While section A, B and D are already realized, implemented and applied successfully, section C is still in progress. Therefore, the next step of future work includes the change of semi-automated code generation to a fully automated code generation process. In addition, to deal with safety and security flaws completely further action is required after the identification. Therefore, future work should include also the assessment of design flaws to evaluate the impacts or severity in detail for diverse quality attributes. Reference [19] already designed a first approach to assess model designs. For this purpose, the approach will be adapted and extended in the future. In addition, to simplify the analysis of already existing IoT systems and the extraction of their models, architecture mining can be considered.

In this paper, we presented an approach to address the issue of safety and security vulnerabilities in IoT systems. As IoT often is applied in safety- and security-critical systems, like medical smart homes, the system must be optimized in the design phase already. Therefore, we introduced our pattern recognition framework which enables the preservation of design knowledge and the structured and automated way to identify flaws in models. The framework consists of four steps. First, the patterns and anti-patterns, which represent desirable or avoidable design, were defined in structured parts. There are parts for generic details, for safety- or security specific information of challenges or assessments and for implementation details to consider the concrete elements and their conditions. Second, a domain specific language was used to realize the defined patterns and anti-patterns. To enable the automated detection of these patterns within the IoT model, a code generation happened in the next step. The last step of our framework represented the identification of vulnerabilities through the defined pattern recognition services. The applicability of our approach was evaluated by an AAL use case. Therefore, we were able to identify two vulnerable design decisions of the IoT system before threats could occur.

Acknowledgment

Electronic Component and Systems for European Leadership (ECSEL) supported the development of this approach within the project CPS4EU (Grant Agreement Number 826276).

References

1. S. R. Department, “Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025,” 2020. [Online]. Available: https://bit.ly/38TUYgO. [Accessed: 25-Feb-2020].
2. G. A. N. Gang and L. U. Zeyong, “Internet of Things Security Analysis,” 2011.
3. J. Viega and G. McGraw, Building Secure Software: How to Avoid Security Problems the Right Way. (Addison-Wesley Professional Computing Series). 2011.

4. J. Rauscher and B. Bauer, “Safety and security architecture analyses framework for the internet of things of medical devices,” 2018 IEEE 20th Int. Conf. e-Health Networking, Appl. Serv. Heal. 2018, pp. 3–5, 2018.
5. RAPID7, “HACKING IoT: A Case Study on Baby Monitor Exposures and Vulnerabilities,” 2015. [Online]. Available: https://bit.ly/1JC9jfS. [Accessed: 24-Feb-2020].
6. FDA, “Cybersecurity Vulnerabilities Identified in St. Jude Medical’s Implantable Cardiac Devices and Merlin@home Transmitter: FDA Safety Communication,” 2017. [Online]. Available: https://bit.ly/2qqkgiA. [Accessed: 18-Feb-2020].
7. M. Zhang, T. Hall, and N. Baddoo, “Code Bad Smells: a review of current knowledge,” J. Softw. Maint. Evol. Res. Pract. 23.3, pp. 179–202, 2011.
8. M. Zanoni, F. Perin, F. A. Fontana, and G. Viscusi, “Pattern detection for conceptual schema recovery in data-intensive systems,” J. Softw. Evol. Process, 2014.
9. P. A. Wortman, F. Tehranipoor, N. Karimian, and J. A. Chandy, “Proposing a Modeling Framework for Minimizing Security Vulnerabilities in IoT Systems in the Healthcare Domain,” pp. 185–188, 2017.
10. W.-T. Lee and P.-J. Law, “A Case Study in Applying Security Design Patterns for IoT Software System,” pp. 978–1, 2017.
11. J. Rauscher, B. Bauer, and M. Langermeier, “Characteristics of Enterprise Architecture Analyses,” Proc. Sixth Int. Symp. Bus. Model. Softw. Des., pp. 104–113, 2016.
12. T. Sommestad, M. Ekstedt, and P. Johnson, “Combining defense graphs and enterprise architecture models for security analysis,” Conf. EDOC 2008, pp. 349–355, 2008.
13. M. Ekstedt and T. Sommestad, “Enterprise architecture models for cyber security analysis,” 2009 IEEE/PES Power Syst. Conf. Expo., pp. 1–6, 2009.
14. P. Johnson, R. Lagerström, P. Närman, and M. Simonsson, “Extended influence diagrams for enterprise architecture analysis,” EDOC, pp. 3–12, 2006.
15. EMF, “Eclipse EMF,” 2020. [Online]. Available: https://bit.ly/3bUNnjU. [Accessed: 24-Feb-2020].
16. R. Gronback, Eclipse modeling project: a domain-specific language (DSL) toolkit. Pearson Education, 2009.
17. Xtend/Xtext, “Xtend/Xtext,” 2020. [Online]. Available: https://bit.ly/2HQcaIc. [Accessed: 30-Jan-2020].
18. F. Carrez, “Internet of Things – Architecture IoT-A Deliverable D1.5-Final architectural reference model for the IoT v3,” no. 257521, 2013.
19. P. Lohmüller, J. Rauscher, and B. Bauer, “Failure and Change Impact Analysis for Safety-Critical Systems: Applied on a Medical Use Case,” Lect. Notes Bus. Inf. Process., 2019.
20. Microsoft, “Microsoft STRIDE,” 2007. [Online]. Available: https://bit.ly/37SKuwE. [Accessed: 17-Feb-2020].
21. M. J. Covington and R. Carskadden, “Threat implications of the Internet of Things,” Int. Conf. Cyber Conflict, CYCON, pp. 1–12, 2013.
22. D. R. Wallace and R. D. Kuhn, “Failure Modes in Medical Device Software: an Analysis of 15 Years of Recall Data,” Int. J. Reliab. Qual. Saf. Eng., vol. 08, no. 04, pp. 351–371, 2001.
23. Department of Defense USA, “Department of Defense Standard Practice System Safety Amsc,” Mctechsystems.Com, no. February 2000, pp. 1–98, 2012.
24. IEC, “Standard 62304, Medical device software — Software life cycle processes,” 2006. .
25. Microsoft, “Microsoft Azure IoT Reference Architecture,” pp. 1–61, 2018.
26. A. Dohr, et al. "The internet of things for ambient assisted living." 2010 seventh international conference on information technology: new generations. IEEE, 2010.
27. G. Pires, et al. "VITASENIOR-MT: a telehealth solution for the elderly focused on the interaction with TV." 2018 IEEE 20th Healthcom, IEEE, 2018.