IoT4CPS – Trustworthy IoT for CPS

FFG - ICT of the Future
Project No. 863129

Deliverable D3.1
Design & Methods Concept

The IoT4CPS Consortium:
AIT – Austrian Institute of Technology GmbH
AVL – AVL List GmbH
DUK – Donau-Universität Krems
IFAT – Infineon Technologies Austria AG
JKU – JK Universität Linz / Institute for Pervasive Computing
JR – Joanneum Research Forschungsgesellschaft mbH
NOKIA – Nokia Solutions and Networks Österreich GmbH
NXP – NXP Semiconductors Austria GmbH
SBA – SBA Research GmbH
SRFG – Salzburg Research Forschungsgesellschaft
SCCH – Software Competence Center Hagenberg GmbH
SAGÖ – Siemens AG Österreich
TTTech – TTTech Computertechnik AG
IAIK – TU Graz / Institute for Applied Information Processing and Communications
ITI – TU Graz / Institute for Technical Informatics
TUW – TU Wien / Institute of Computer Engineering
XNET – X-Net Services GmbH

For more information on this document or the IoT4CPS project, please contact:
Mario Drobics, AIT Austrian Institute of Technology, mario.drobics@ait.ac.at

Document Control

Title: Design & Methods Concept
Type: Public
Editor(s): Stefan Jaksic
E-mail: Stefan.Jaksic@ait.ac.at
Author(s): Christoph Schmittner (AIT), Denise Ratasich (TUW), Martin Matschnig (Siemens), Lukas Krammer (Siemens)
Doc ID: D3.1

Amendment History

Version	Date	Author	Description/Comments
V0.5	09.11.2018	Christoph Schmittner	Initial version prepared
V0.6	05.03.2019	StefanJaksic, Christoph Schmittner	
V1.0	07.03.2019	Stefan Jaksic, Christoph Schmittner	Final version

The IoT4CPS project is partially funded by the "ICT of the Future" Program of the FFG and the BMVIT.

Federal Ministry
Republic of Austria
Transport, Innovation
and Technology

FFG
Forschung wirkt.

Content

Abbreviations

CPU	Central Processing Unit
DFD	Data Flow Diagram
EA	Enterprise Architect
FMVEA	Failure Mode, Vulnerabilities and Effects Analysis
I4.0	Industry 4.0
IoT	Internet of Things
IIOT	Industrial IoT
QoS	Quality of Service
ROTS	Real-Time Operating System
SHSA	Self-Healing through Structural Adaptation
UC	Use Case
V&V	verification and validation
WCET	Worst Case Execution Time
WP	Work Package

Executive Summary

In the future it is not sufficient to interlink safety, security in the operational phase of a system. For a dependable IoT system, all dependability attributes must be considered not only during the operational phase, but also during installation and commissioning. Thus, engineering tools and novel theoretical concepts are required to increase dependability of IoT devices for Cyber-Physical Systems (CPS). This is specifically challenging due to the heterogeneity of different technologies used in IoT and CPS.

In this deliverable we focus on interconnecting already ongoing engineering activities at multiple levels and enrich them with tools to address dependability aspect. This deliverable outlines theoretical and practical contributions in different layers of CPS architecture. We first report on our tool for standard-based product development management: GSFlow. We introduce Failure Mode, Vulnerabilities and Effects Analysis (FMVEA) as a novel approach to integrated safety and security analysis. In order to further support security requirements analysis, allocation, and management we focus on the Model-based Security Requirement Management Tool: MORETO.

In order to allow for dependability, the developers have to cope with ever-growing landscape of different technologies used in modern IoT architectures. To overcome this problem, we propose a recommender system for dependable IOT applications. Finally, as a mean to improve the fault-tolerance of IoT systems we advocate for Self-Healing by Structural Adaptation.

1. Introduction

D3.1 is the first deliverable in WP3, which collects and presents potential solutions, tools and methodological building blocks for the development of safe and secure IoT and CPS systems. This deliverable puts a special focus on the integration of privacy and on the support of the complete engineering cycle, from engineering support to providing potential solutions.

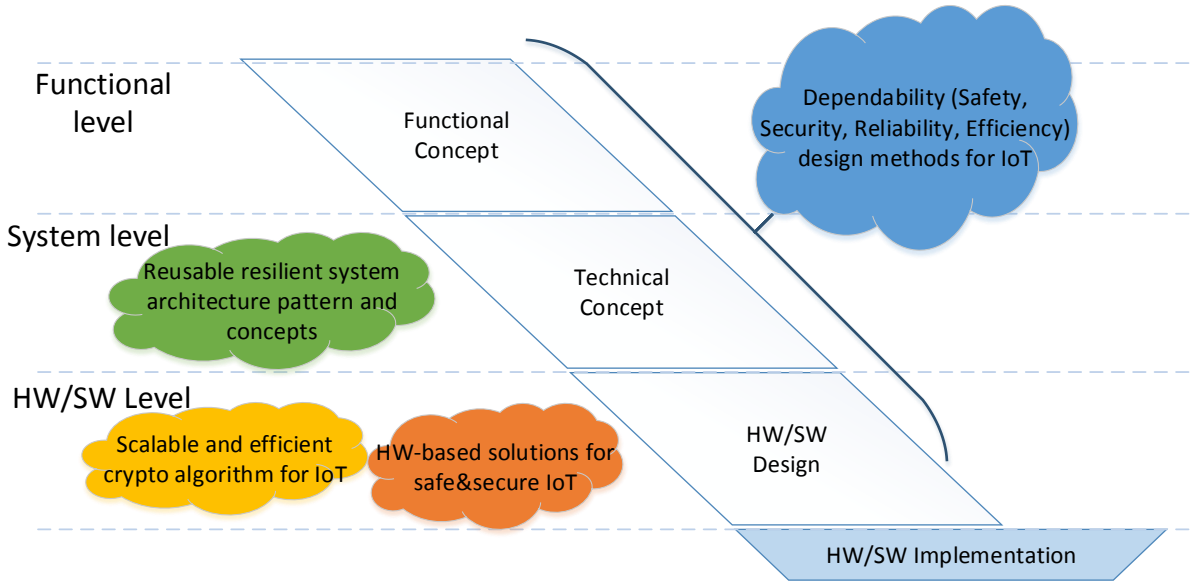

Figure 1: Structuring WP3 contributions along the V-Model

The contributions are structured based around the left side of the V-Model. The usage of the V-Model is here mainly as a structuring model and for easier classification and explanation. There exist multiple engineering process models and some might be better suited. Therefore, the usage of the V-Model should not be understood as an enforcement of this process model and the presented methods and building blocks are not restricted in term of engineering process model.
Our understanding of Dependability itself is based on (Avižienis, Laprie, & Randell, 2004). In this work, the Dependability is differentiated between Attributes, Threats and Means (please see Figure 2).

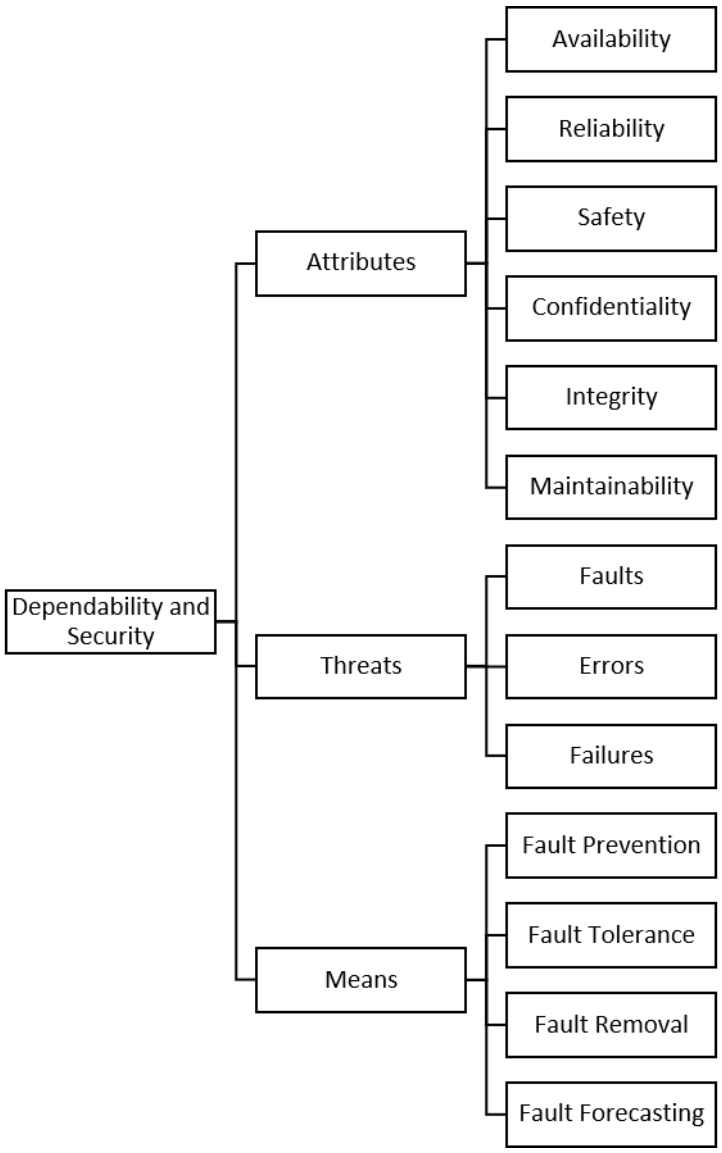

Figure 2: Dependability Tree

The Attributes summarize all parts of Dependability, e.g. the set of properties we would like to ensure that a system we need to rely on possess. Although IoT4CPS project sets priority on safety and security, the other attributes are also relevant.

Threats are potential factors which can cause a violation or contribute to a violation of a Dependability Attribute. In order to protect the Attributes, we can use different Means.

The following section will give an short overview about the basic system concept considered in IoT4CPS and present then different Means to achieve dependability, which are considered in WP3 of IoT4CPS project.

2. Overview and System Model

Various applications used by the system can be implemented in a modular fashion, as services. The services are also used to implement procedures which are responsible for safety monitoring, system diagnostics, behavior learning, optimization and system recovery.

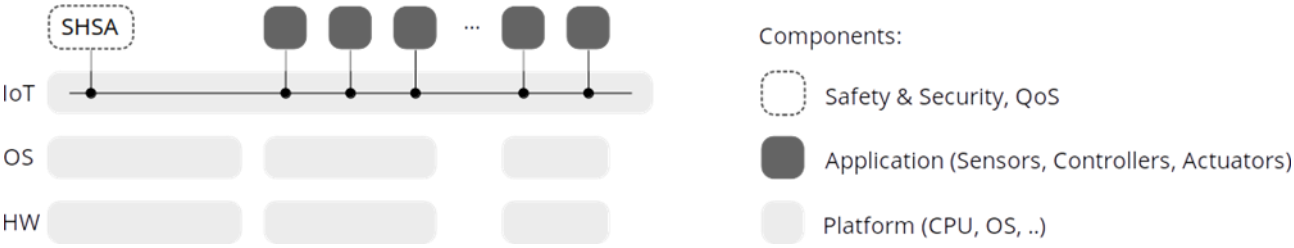

Figure 3: Architecture of a distributed CPS using the IoT as communication backbone.

A realization of this abstract model will be implemented in the demonstrator (see D6.1 for an example).

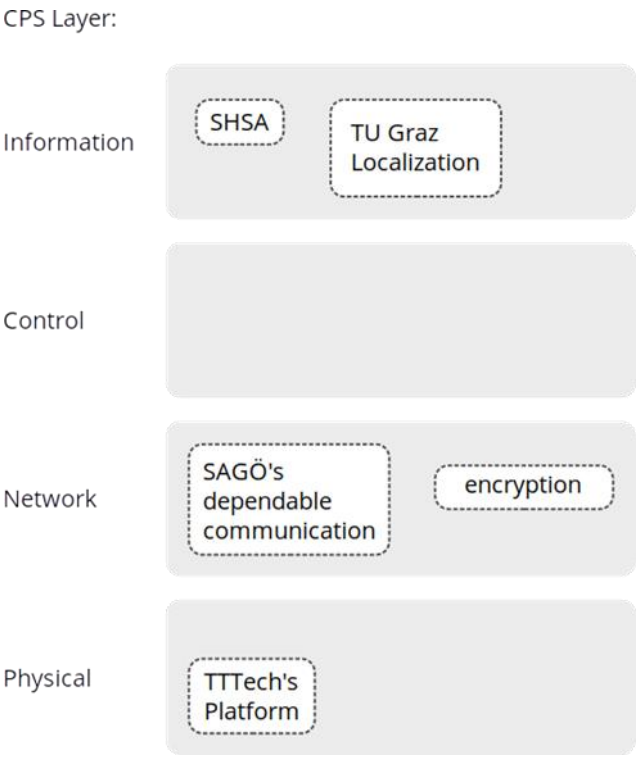

Figure 4: Contributions to safety and security can be separated into different CPS layers.

3. Contributions

In this section we describe the contributions of D3.1. We provide methods, tools and architectures for safe and secure IoT for CPS.

3.1 GSFlow

Our first contribution is a tool for standard-based product development management: (GSFlow). It is one of the results of a more general effort to develop tools to support model-based development approaches and Safety & Security by Design.

3.1.1 Overview

The goal of GSFlow is to support the complete engineering lifecycle of safety and/or security relevant systems based on pre-defined processes, by guiding the user through the development process. Its main objective is to make standard driven development straightforward, especially for companies that are unacquainted with functional safety and security standards. The central output of GSFlow is a safety and/or security case which shall support companies two ways: (1) helping to reach assurance for their products and (2) allowing to summarize the argumentation why a system is acceptably safe and secure. To achieve this goal every project in GSFlow contains Requirements which correspond to functional safety and security standards. GSFlow provides standard conformant user management and utilizes tools to ensure the quality of an evidence.
GSFlow provides a flexible framework for modelling and executing standards. It is also capable of executing plugins written by an external developer. This flexibility ensures that the specific needs of a company can be met. Furthermore, an external developer is only required to implement an interface according to their needs. For example, GSFlow can execute plugins for Report Generation, for checking the artifacts and requirements using Natural Language Processing methods and for executing the tests.

3.1.2 GSFlow Structure and Definitions

In this section we provide more details about the operation of GSFlow.

Requirement

Requirements are defined as the entities needed to achieve the objective of the project. Two different kinds of Requirements can be distinguished.
The first kind of requirements are the Standard Requirements, which are derived from functional safety and security standards. They are needed to identify the goals that are obligatory to reach compliance to relevant standards. Secondly, the Product Requirements in GSFlow represent the requirements that are specific to a product. They can be linked to a Standard Requirement. Furthermore, Requirements in GSFlow are the key elements for documenting the workflow as they serve as an anchor to attach evidence and trace every action that is conducted on them while being processed. Once processing is completed and all evidence has been created, a requirement may enter the state of completion.

Phase

Phases in GSFlow represent phases of a safety/security standard. GSFlow ensures that in every phase Standard Requirements that are specific to this phase need to be fulfilled. A phase can only start once all previous phases have been completed, with an exception for parallel phases which can always be refined during the whole development lifecycle. Phases in GSFlow are used to structure Requirements and define the order of execution in the workflow.

Tools and Plugins

A tool represents an external application that can be plugged into GSFlow. By implementing and using the interfaces provided by GSFlow, an external Developer can create plugins and an admin can upload them to GSFlow after conducting validation. These tools/plugins are then ready to be executed. The dataflow between the GSFlow and the tool/plugin is conducted only through the API. The utilization of tools shall serve as quality assurance as well as provide convenience to the users working with GSFlow.

User management

GSFlow supports standard conformant user management. Different roles can be assigned to different users per project. Roles are based on a generic model of roles defined in different standards. In GSFlow, roles can be mapped to specific standards. The list of the supported roles include:

- Project Manager
- Requirements Manager
- Developer
- Verifier
- Validator
- Assessor

3.2 FMVEA

Failure Mode, Vulnerabilities and Effects Analysis (FMVEA) is based on the Failure Mode and Effect Analysis (FMEA) and extends the standard approach with security related threat modes [23]. First a system is modeled. Then the failure and threat modes for each element of the system model are identified. While a failure mode describes the way the function of an element fails, a threat mode describes the way in which the identified function of an element can be misused. Threat modes classifies threats in six categories (Spoofing of user identity, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege). Depending on the domain, the system architecture and the knowledge about the system, failure and threat modes can be refined and extended. Each identified failure or threat mode associated with the element is investigated for potential effects. For modes with critical effects, potential causes are analysed and the likelihood for each cause is estimated. For threat modes, likelihood is determined using a combination of threat and system properties. Threat properties mainly describe the resource and motivation of a potential threat agent while system properties include reachability and system architecture. The system model is based on a three-level data flow diagram (DFD). Effects of failure and threat modes are presented at the context level of the diagram, which shows the interaction between the system and its environment. Failure and threat modes are located at the level 1 DFD. Vulnerabilities and failure causes are based on the level 2 DFDs.

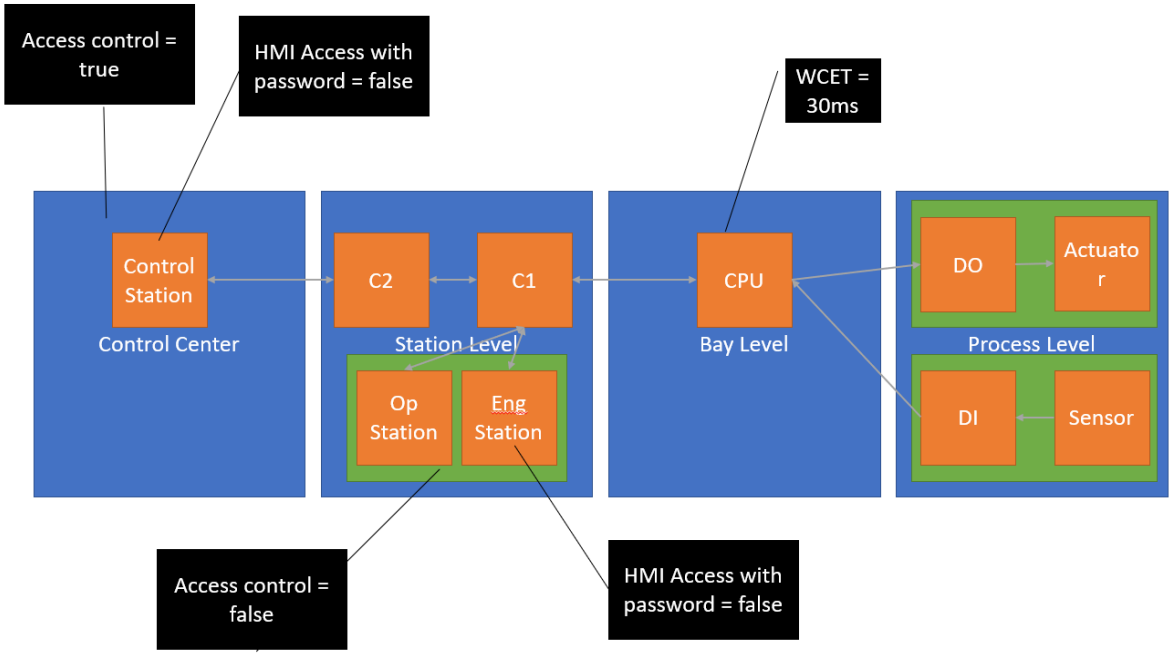

Figure 5: An example model as an input for analysis with FMVEA

Figure 5 shows the diagram of an example use-case in which a set of actors is controlled by an CPU, based on some sensor input. Control Strategy can be defined and monitored in some higher layers. The example model will be used to perform an analysis with FMVEA. The blue squares represent the root environments of a specific level. Each orange node can be seen as an operating element inside the environment. Green squares represent sub-environments inside the root-environments which encapsulate their own operating elements. Black squares display the defined attributes/properties for the diagram elements. For example, the “CPU” has the attribute/property “WCET” (“Worst Execution Time”) with a value of “30 ms”.

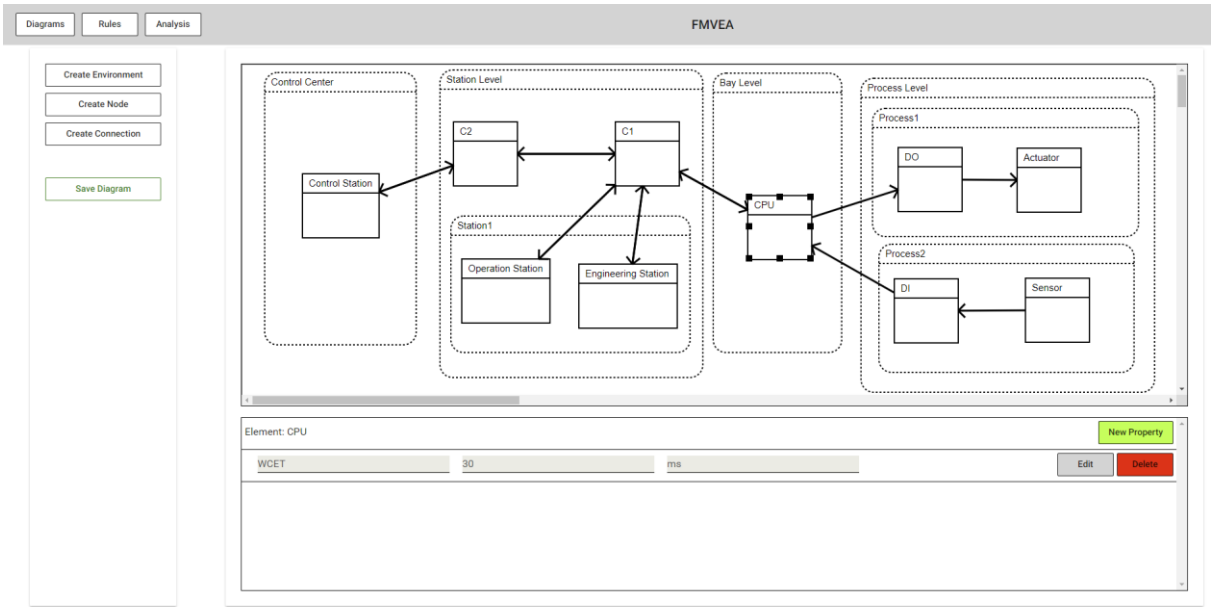

Figure 6: The diagram modeled in FMVEA

Figure 6 shows the diagram from Figure 5 modeled in FMVEA. On the left-hand side, we can see the diagram related actions like “Create Environment”, “Create Node” and “Create Node”. An

Environment can be considered as a container, which provides general attributes to his children. Attributes can be focused on Security and Safety. The attributes of an element are directly displayed below the diagram.

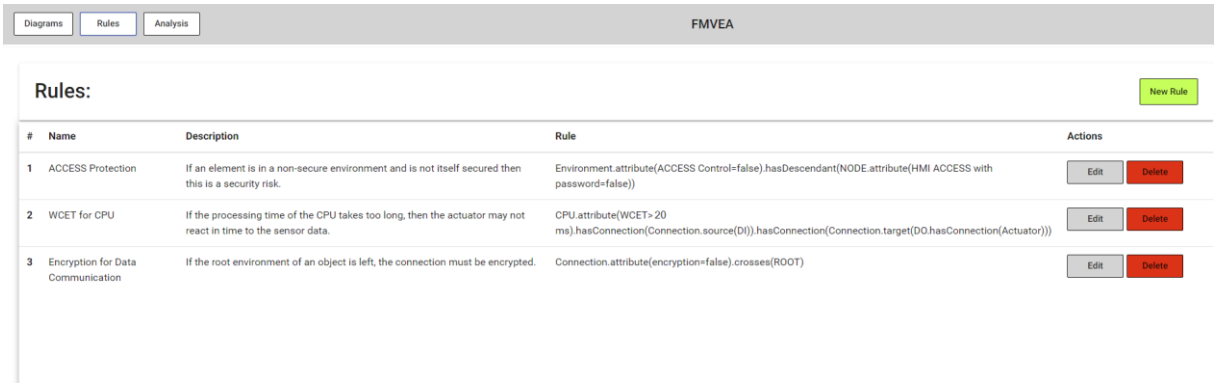

Figure 7: Rules defined for the Use case

Figure 7 displays the rules which should be used to analyze the use-case diagram. From the left to the right are the names of the rule then a short description and the “Rule”. The “Rule”-column is the most important one, because here the actual rule for the analyzer is defined. The content of a rule is defined by the grammar shown in Figure 8.

```
Expression → ( ObjExpression | ConnectionExpression )
ObjExpression → ( SingleObjExpression | MultipleObjExpression )  AttributeFilter:? ConnectionFilter:? RelationFilter:*
SingleObjExpression → Identifier
Identifier → singleString
MultipleObjExpression → "{"  ( AndConnectedObjects | OrConnectedObjects ) "}"
AndConnectedObjects → "{" ObjExpression _ "AND" _  ObjExpression  ( _ "AND" _ ObjExpression ):* "}"
OrConnectedObjects → "{" ObjExpression _ "OR" _  ObjExpression  ( _ "OR" _ ObjExpression ):* "}"
AttributeFilter → ".attributes{" Attribute "}"
Attribute → (SingleAttribute | MultipleAttribute)
SingleAttribute → ( StringValueAttribute | NumericValueAttribute | MultipleStringValueAttribute| MultipleNumericValueAttribute)
MultipleStringValueAttribute → multipleString "IN {" singleString:+ "}"
MultipleNumericValueAttribute → multipleString "IN {" ((singleNumber | decimalNumber) (_ attributeUnit):?):+ "}"
StringValueAttribute → multipleString "=" singleString
NumericValueAttribute → multipleString ("=" | ">" | "<" ) (singleNumber | decimalNumber) (_ attributeUnit):?
MultipleAttribute → (AndConnectedAttributes | ORConnectedAttributes
AndConnectedAttributes → Attribute ("," _ Attribute):+
ORConnectedAttributes →  "{" Attribute ( _ "OR" _ Attribute ):+ "}"
ConnectionFilter → ".hasConnection{Connection" (SourceFilter | TargetFilter) CrossesFilter:? AttributeFilter:? "}"
SourceFilter → ".from{" ObjExpression "}"
TargetFilter → ".to{" ObjExpression "}"
CrossesFilter → ".crosses{" ( "PARENT" | "ROOT" ) "}"
RelationFilter → ( ".hasDescendant{" ObjExpression "}" | ".hasAncestor{" ObjExpression "}" )
ConnectionExpression → "Connection" SourceFilter:? TargetFilter:? CrossesFilter:? AttributeFilter:?
singleString → [a-zA-z]:+
multipleString → [a-zA-z]:+ ( _ [a-zA-z]:+ ):*
singleNumber → [0-9]:+
decimalNumber → [0-9]:+ "." [0-9]:+
attributeUnit → [a-zA-z/]:+
```

Figure 8: Example of the FMVEA grammar

Figure 9 displays the results of the use-case analysis. The previously created rules are applied on the selected diagram.

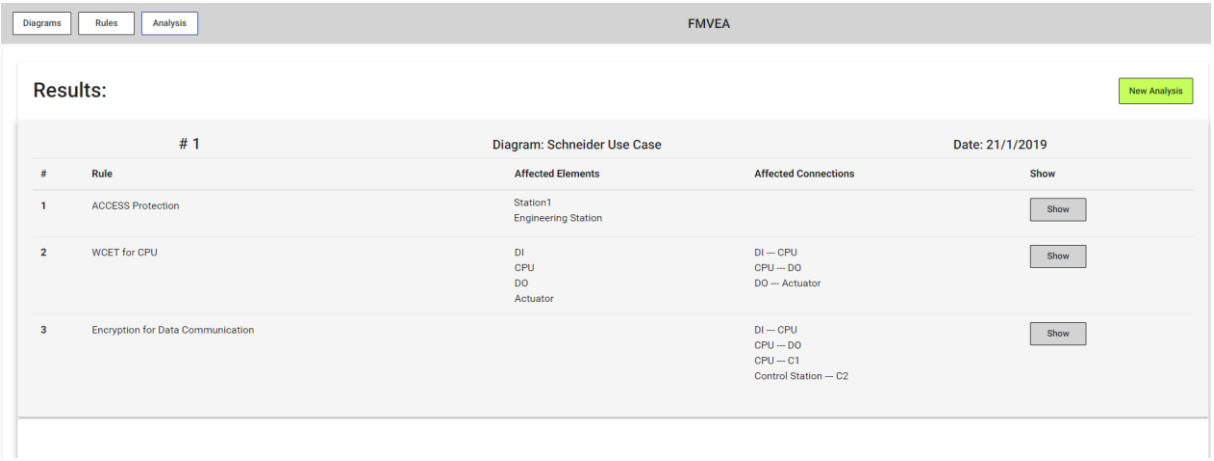

Figure 9: Analysis results for the defined rules and the diagram

From the left to the right you can see the applied rule and the results of the specific rule on the diagram. The affected elements and connections can be viewed in the diagram if the user clicks on the "Show"-button to the right. Inside the diagram the affected elements and connections get highlighted by a red border as you can see in the Figures 10-12.

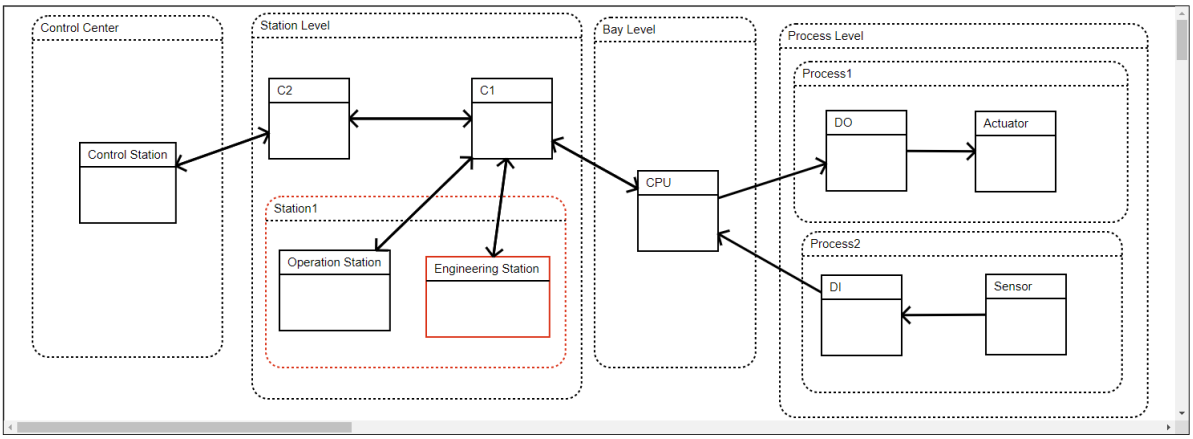

Figure 10: 1st rule results

In Figure 10 you can see the affected elements of the first rule. The Definition of the first rule says that if there is any environment with the property "ACCESS Control=false" which contains a child object with the property "HMI ACCESS with password=false" then there is a security problem. As one can observe from Figure 5 we initially defined the "Control Station" with "ACCESS Control=true" so there is no problem even if the "Control Station" has the property "HMI ACCESS with password=false". However, if one observes the "Station 1" and the "Engineering Station" then both criteria are fulfilled, and this is the reason why these two objects are the affected objects of the first rule.

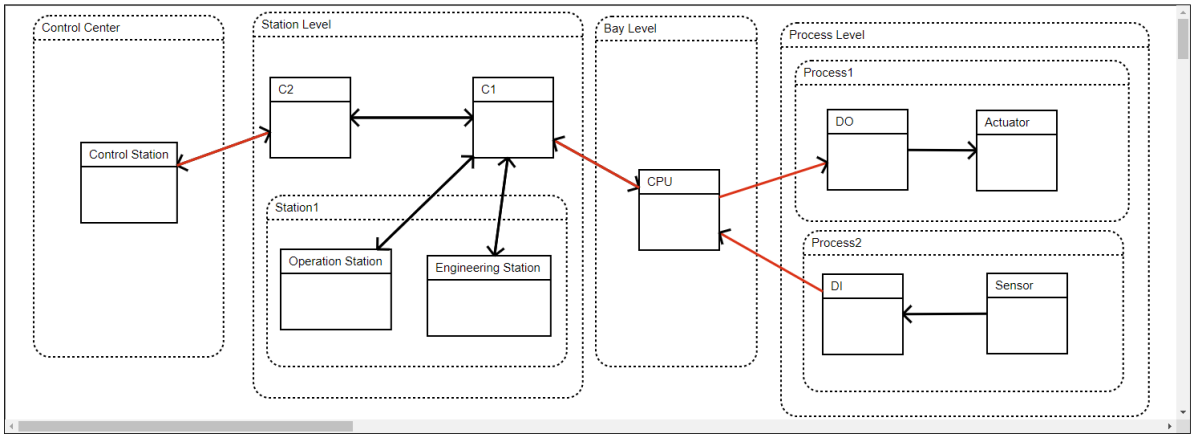

Figure 11: 2nd rule results

In Figure 11 you can see the affected connections of the second rule. The Definition of the second rule says that if there is any connection between two objects which do not share the same root object then the connection must be encrypted. One can observe from Figure 5 we never defined an encryption property for any connection inside the diagram. For example, take the connection between the "Control Station" and "C2". The root object of the "Control Station" is the "Control Center" and the root object of the "C2" is the "Station Level" but they share a connection. Now let's consider the connection between the "C2" and "C1". The root object of both objects is the "Station Level". They share the same root element as the same parent element, therefore this connection does not represent a potential problem.

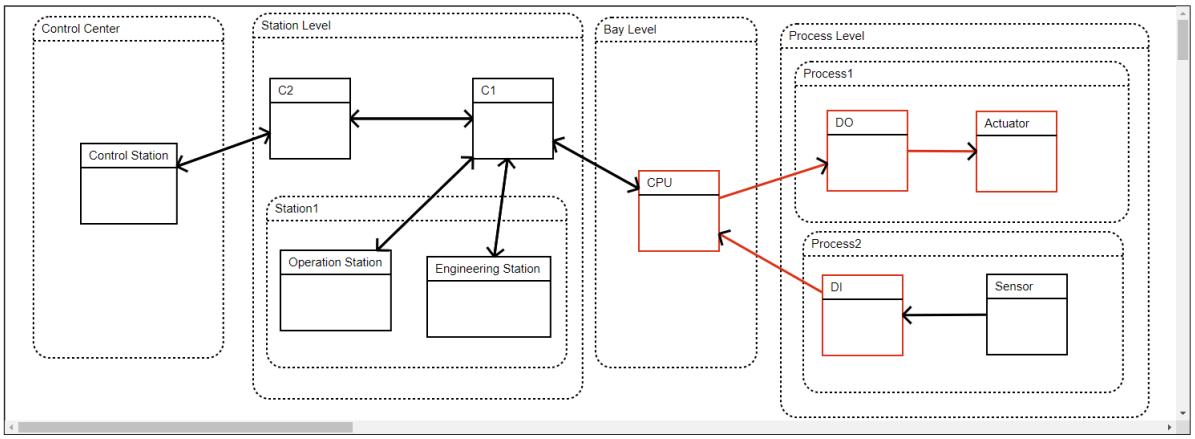

Figure 12: 3rd rule results

In Figure 12 you can see the affected elements and connections of the third rule. As you can take from Figure 5 we set the WCET of the CPU as "30 ms". The Definition of the third rule says that if the WCET inside the CPU is higher than "20 ms" the actuator may not react in time to the sensor data. The sensor data is coming from the "Data Input" (DI) then the data is processed inside the CPU with a WCET of 30ms and then sent to the "Actuator" over the "Data Output"(DO). This is the reason why the "DI", "CPU", "DO" and the "Actuator" are affected elements in this case. The connections between these objects transport the data and are affected connections in this case. This rule could be expanded by additionally taking into account the latency of the connections.

3.3 Threat Modeling

Threat modelling is a technique for the security analysis process. This technique defines an abstract model of potential threats which can be applied to a system model to identify representations of the threats. AIT's threat modelling tool allows to automate this process by formalizing threat information. The threat modeling uses several source materials to ensure a range of threats is considered. Figure 13, depicts the data flow between different components in a vehicle with Roadside unit.

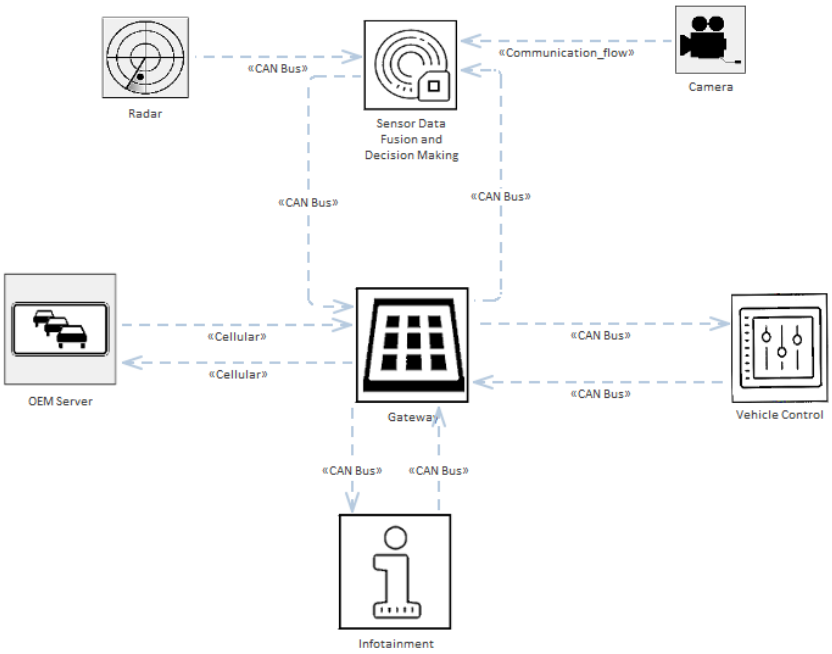

Figure 13: Data flow for threats assessment

Based on the given model, and without any security mitigation measures, GSFlow successfully identified 69 threats. The tool automatically assesses the identified threats to find the potential security issues. The results are displayed in a clear and readable form which allows for traceability of the results, as depicted in Figure 14.

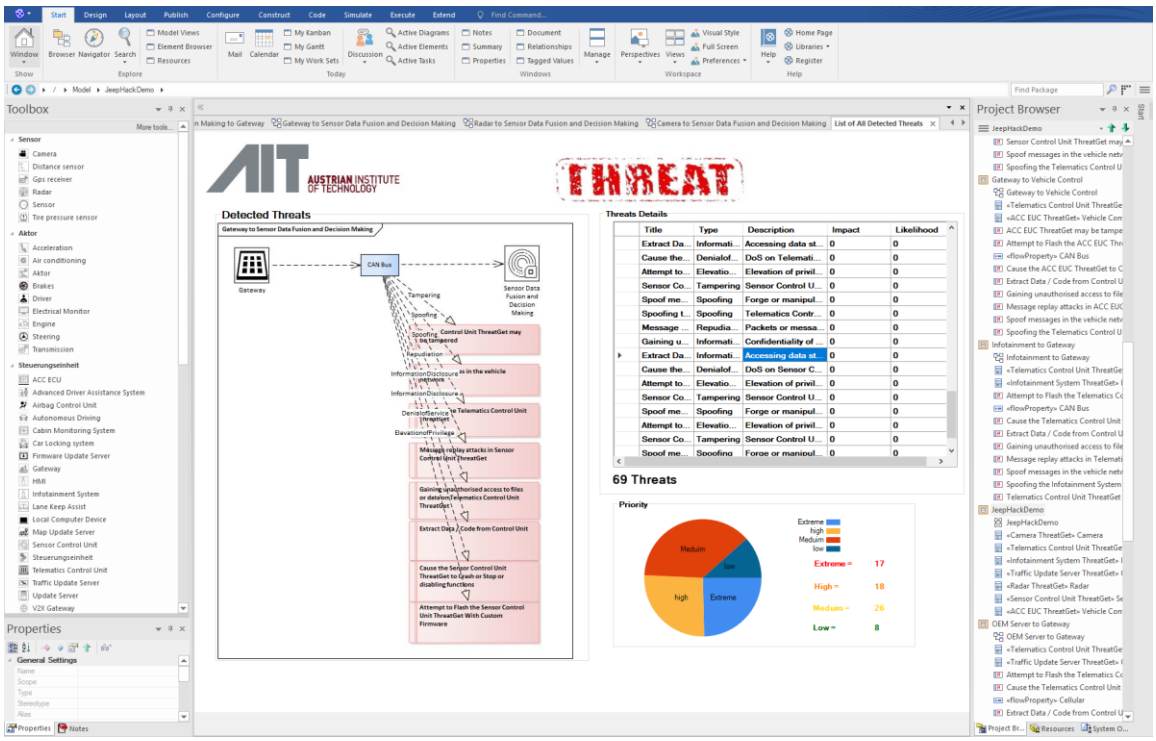

Figure 14 The results of the threat modeling tool

3.4 Moreto

The correct security requirement identification and efficient security requirement management are essential for any security engineering process. We can design, implement, and test a secure system only if we know the exact security requirements. Achieving an efficient requirement management is a challenge in system development. The Model-based Security Requirement Management Tool (MORETO) serves a tool for security requirements analysis, allocation, and management using modelling languages such as SysML/UML. MORETO is an Enterprise Architect (EA) plugin for managing the IEC 62443 security standard. It is a reliable and a flexible to model safety & security requirements suited to different components and system architectures. It generates a list of security requirements in a given diagram, which can help the user to build-up a secure infrastructure. Figure 15 shows a simple example of different components which interact together through a network.

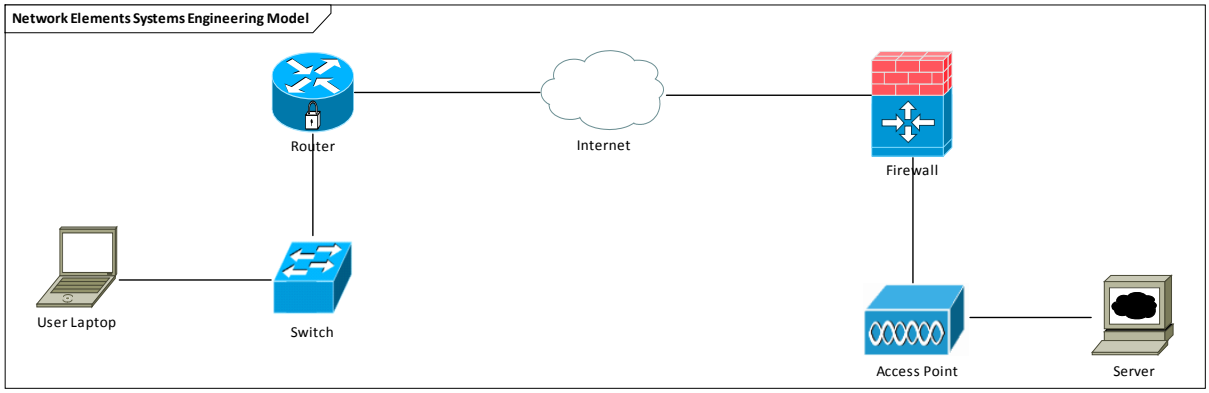

Figure 15 Simple network elements for system engineering model by MORETO

MORETO scans all the elements of a given model and automatically generates a list of security requirements based on expert knowledge encoded in the tool itself and on the description contents of the IEC 62443. Figure 16, shows a list of identified security requirements of the Router and Switch devices respectively.

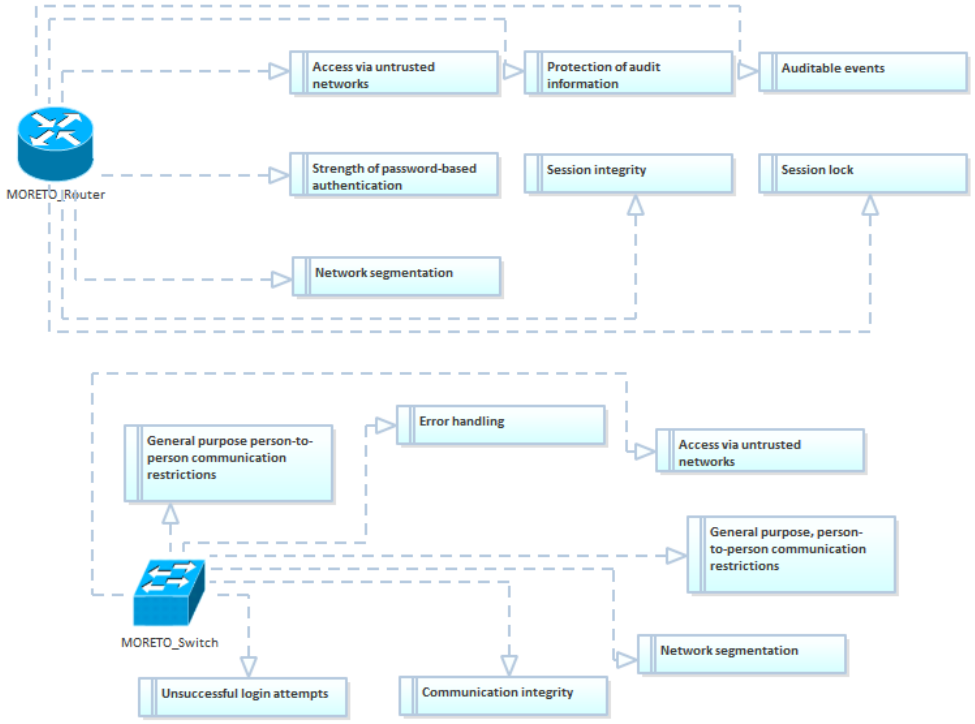

Figure 16 IEC 62443 Security standards for router and switch devices

3.5 A recommender system for dependable IOT applications

3.5.1 Motivation

Along with the growing market of Industrial IoT (IIOT) applications, the set of available network technologies is continuously expanding. Today, developers have a huge set of connectivity networks at their disposal, ranging from short-range networks such as Bluetooth to global connectivity via satellite networks [1]. Figure 17 tries to give an overview of existing technologies while not claiming to be exhaustive. Depending on the specific use case of each IIOT application, different approaches constitute the most cost-effective network technology solution, as there is no “one-size-fits-all” solution. Choosing the set of network technologies, which fits the needs of the IIOT use case must be a careful trade-off between the ability of the technologies to meet specific functional requirements and the related costs [1]. Complex IIOT systems have a significantly larger set of dependability requirements compared to “normal” IoT applications. As the systems connectivity network plays a major role in fulfilling these requirements, choosing the correct set of technologies is crucial.

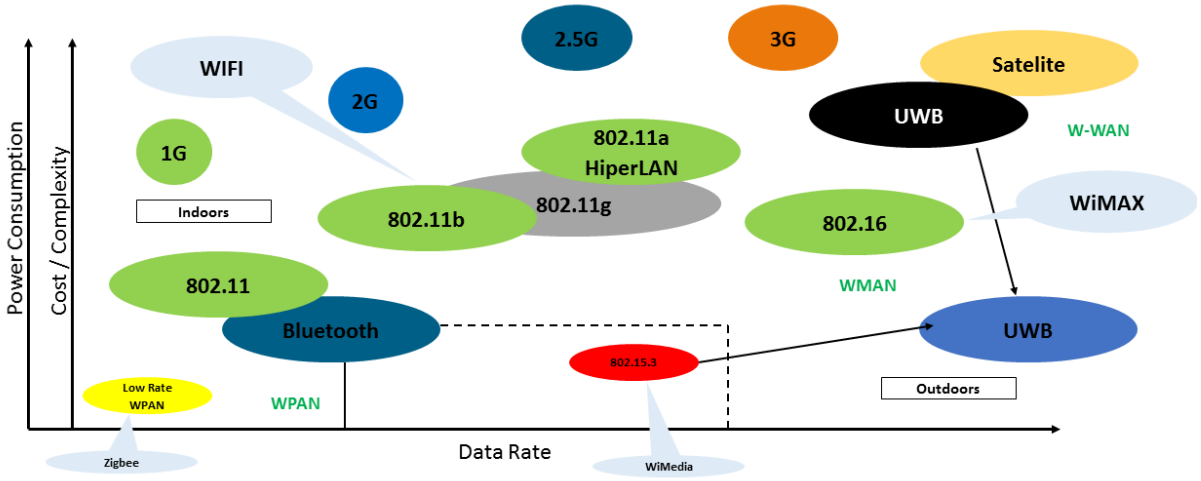

Figure 17:Technology overview [1], [2], [3], [4]

3.5.2 Aim

Depending on the application and its scope, different system architectures are required. However, most applications follow the generic system architecture depicted in Figure 2. This architecture comprises different system levels and possible interconnections.

The challenge in building such systems lies on interconnecting already ongoing engineering activities and brownfield devices at multiple levels and enrich them with tools to address dependability aspects of the system. Usually such a system design is derived by experts, here another possibility will be explored, i.e. building a recommender system that relates the application specific requirements (e.g., purpose, location, connectivity, power supply) with properties of different technologies (e.g., energy consumption, communication bandwidth) – resulting in a feasible system setup (system topology, technologies to apply within the system) regarding a specified use case, without the need of consulting experts.

Beside generating a suggested system topology, the recommender system will match the application's demand regarding dependability attributes with the offered attributes of the possible technologies. Thereby, qualitative and quantitative measures are applied. Based on this, a final selection of technologies is possible.

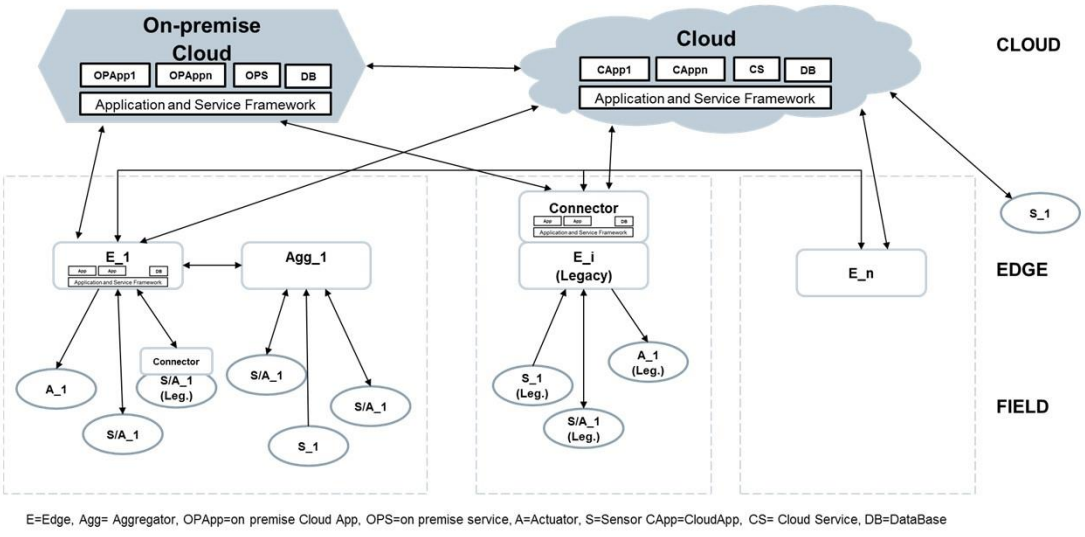

Figure 18: Generic System overview

3.5.3 System Overview

This ability is achieved by maintaining a database, comprising of the technical and functional characteristics of all the available network technologies. The underlying implementation will filter off the technologies which does not match the user specifications, leaving the most suitable for the user's application. As stated earlier, the capabilities of the recommender system will be focus mainly on wireless IoT technologies. In this domain, it can be basically distinguished between three operational models [3]: a) Bluetooth model where intending users are expected to handle both the purchase of IoT equipment as well as connectivity issues. b) Wifi model where consumers are expected to subscribe to the services of a local network provider. These networks basically operate on the license exempt spectrum. c) Cellular Operator Model is fundamentally aimed at consumers that require global connectivity. Technologies in this category are more reliable since they operate in the licensed spectrum.

The recommender system is intended to suggest a feasible combination of wireless communication technologies (or services) based on the user's specifications. Figure 19 gives an impression on the user interface of the introduced Recommender System and gives an example of possible input parameters. Based on these parameters, it generates a template for a system topology, recommends a selection of technologies including key parameters, visualizes the results and provides additional information such as dependability properties.

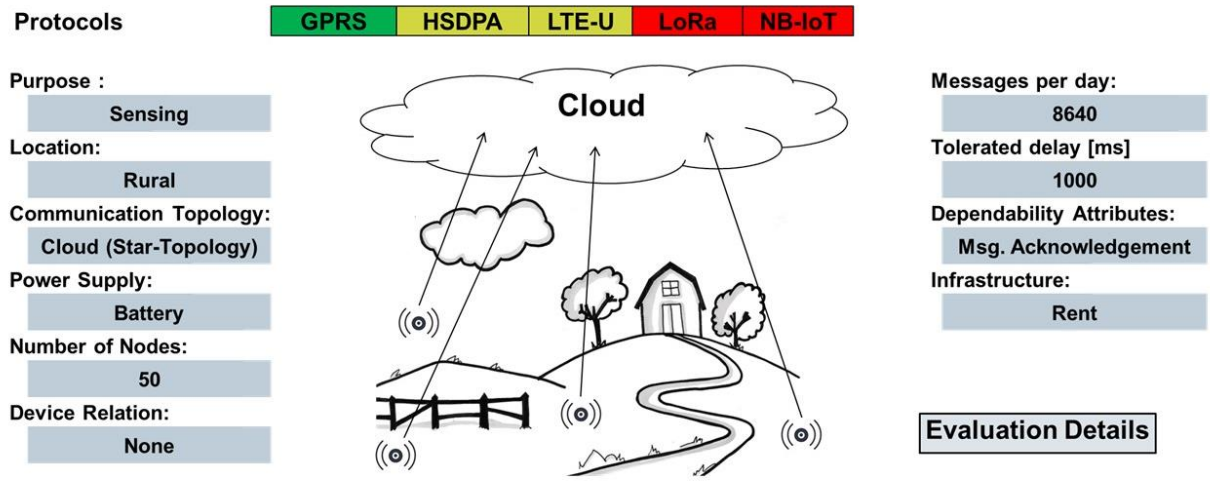

Figure 19: Illustration of the Recommender System with possible input parameters

3.5.4 Network technologies and evaluation criteria

The technical characteristics of the different network technologies are described by various parameters including both protocol-based and deployment-based properties. In order to evaluation if a given technology is suitable for the implementation of a specific use case protocol properties and criteria can be used. This section outlines criteria which can be used to assess the applicability of a given technology regarding a specific use case and can thus be used as a starting point for the implementation of a recommender system. The categories were established based on an initial literature review [1-22].

3.5.5 Protocol Properties

Coverage [Global / Rural / Urban / Local]

States the geographical spread of the IIOT application and the therefore needed spatial range of the radio link. This will either lead to the deployment of a short-range network such as Bluetooth or a long-range network such as NB-IoT.

Max throughput Up / downlink [B/s]

The maximum throughput needed for both Up- and Downlink between transceivers in the desired IIOT system limits the possible set of network technologies. Always using a technology with a high data rate might have a negative impact on the energy consumption, so an adequate estimation of the throughput is necessary.

Max number of messages

Beside the maximum data rate, the overall amount of transferred data influences the selection procedure. Especially when subscribe to the services of a local network provider, the number of messages and the overall data volume directly affects the resulting costs. In addition, dedicated IoT protocols (such as NB-IoT and LoRA) might exhibit a maximum number of daily messages thus limiting the application of such technologies in message rich application contexts.

Worst-case latency

Beside the need of low latency technologies for real-time applications, different IIOT use cases might require different worst-case latency regarding its underlying connectivity network. Depending on the use case the tolerated worst-case latency can range from minutes or seconds (e.g. detecting cars on a parking lot) to (sub-)milliseconds (e.g., real-time system monitoring / control or time critical systems (safety applications)).

Real-time capability

Most IoT solutions are geared towards real time applications thus, an end node in such a solution must run a real time operating system (RTOS) in order to process data without delays. This characteristic thus denotes the ability of the underlying network technology to support the demands of an RTOS by meeting strict latency requirements to avoid system failures.

Data Delivery Constraints

Different IIOT applications can have different QoS requirements which must be reflected in the underlying network technologies. This characteristic thus denotes the readiness of a technology to behave in a deterministic way and therefore its ability to meet the QoS requirements of the envisioned application.

Security Overhead

This category evaluates the costs associated with securing a specific network technology. This is done by factoring in a technologies ability to deal with heterogeneity, hostile environments and unreliable data communication.

3.5.6 Deployment Properties

Infrastructure Deployment Cost and Constraints

There are two types of IoT networks [22]: Infrastructure and Adhoc based networks, which are typically deployed using different network topologies. The “Infrastructure Deployment Cost and Constraints” characteristic tries to classify standards based on their operational models, with three potential operational models being considered:

Rented - For standards which are infrastructure based and are managed with the Cellular Operated Model.

Rented (Monopoly) - For standards which are infrastructure based and are managed with the WiFi Model.

Owned - For standards which are either infrastructure or ad-hoc based but are managed with the Bluetooth Model. The Monopoly-version of this category simply denotes that the standard is partially proprietary.

Environment Constraints and Robustness

Different applications have different tolerance levels regarding interruptions from the propagation medium as well as man-made causes. This characteristic tries to classify or rank the network technologies based on their tolerance (reliability) to unforeseen failures.

Energy Consumption

This category evaluates the energy consumption associated with each connectivity standard, as energy consumption plays a critical role for deployment.

Interoperability

Since all IoT network connectivity technologies are packet based and most application requirements are uplink intensive, interoperability can be defined as the ability of an IoT node to transverse all sorts of networks without being scrambled or corrupted (via any form of gateway translations).

Scalability

Scalability of the network technologies in IIOT systems is a critical issue due to the high number of nodes and a possibly high node density. Some applications might require to forsee their future expansion. Relevant technology parameters for achieving a certain level of scalability are for

example the maximal numbers of connected nodes, the integration effort for adding new nodes or the scalability of routing protocols.

System Form Factor

This category denotes the conceivable size of a node in a particular technology. Since the small form factor of NFC tags hugely depends on its antenna size as well as ability to function passively. It was decided to tentatively evaluate these characteristics based on the operating frequency of the technology in consideration since it determines the antenna size. Therefore, lower frequencies imply higher system form factor while higher frequencies imply lower system form factor.

3.5.7 Next steps

In a first step the categories were defined. In a next step the technologies identified in (see section 1.1) will be categorized regarding the proposed categories. The categorization will be established in a machine-readable format to allow establishing the proposed recommender system.

3.6 Self-Healing by Structural Adaptation

Failed observation data may compromise system's safety. For instance, an erroneous detection of surrounding vehicles that is input to the path planning unit of an autonomous vehicle might cause fatal consequences. Traditional fault-tolerance is aims to overcome such critical failures.
Self-healing can be applied to react also to failures not specifically considered during design-time, e.g., faults caused by functional, environmental or technological changes or zero-day malware. A very promising approach of achieving *self-healing* is through *structural adaptation* (SHSA), by replacing a failed component with a substitute component by exploiting implicit redundancy (Fig. 20) [24][25].

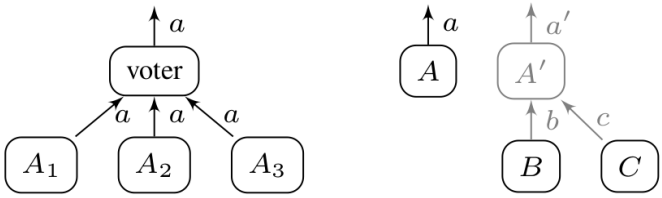

Figure 20: Types of redundancy - explicit (left) and implicit (right) [RHISG2017].

In particular, SHSA can be used to replace failed observation data. It monitors and substitutes CPS variables (cf. signals) in messages communicated between application components (e.g., sensors and controllers) based on a knowledge base modeling the relations between the variables.
SHSA can be encapsulated in one or more components listening and acting on the communication network of the IoT. The detection identifies a failed component by comparing its output to related information on the network using the relations encoded in the knowledge base. A failed component may be removed or shut-down to avoid faulty messages and possibly propagating the failure. Then SHSA spawns a new component – the *substitute* for the failed one. The substitute subscribes to related information and combines these (again by using the relations in the knowledge base) to provide the needed information.
In this project we specifically target extensions to the knowledge base and substitution, the architectural requirements regarding security and the fault detection. First results are:

- Guided search of a substitution (speed-up of the recovery process). Evaluation and selection of a substitution extracted from the knowledge base [26].
- Architectural requirements for SHSA and considerations w.r.t. security (D6.1).
- Extensions to the knowledge base: encode relations in Prolog (rule-based knowledge base to enable requirements checks). Implement state-aware relations (formerly only state-less relations were possible). Demonstration of how to handle time in relations.
- Fault detection: automatic generation of a runtime monitor for related information considering availability of the information and possible time delays (e.g., latency, physics).

Related work is presented in [27] and D2.1. Case studies and examples can be found in [25],[27] and D6.1.

4. Conclusion

With the service-oriented and dynamic nature of IoT systems ensuring safety and security during the design process is a challenging activity. D3.1 collects a first overview about potential approaches regarding safety & security engineering, methods to:

- Identify and assess risks
- Ensure compliance with standards and best practice guidance's
- Optimize risk treatment decisions

Due to the ever-increasing system complexity it is necessary to support the engineering process to ensure security by design. D3.1 documents a collection of approaches, which can be applied and evaluated in multiple IoT4CPS use cases.

5. References

[1] Frederic Vannieuwenborg, Sofie Verbrugge and Didier Colle. “Choosing IoT-connectivity? A guiding methodology based on functional characteristics and economic considerations”. Transactions on Emerging Telecommunications Technologies(2018;29:e3308)

[2] Elkhodr, Mahmoud, Seyed A. Shahrestani, and Hon Nin Cheung. "Emerging wireless technologies in the internet of things: a comparative study." International Journal of Wireless and Mobile Networks 8.5 (2016): 67-82.

[3] Tardy, Isabelle, et al. "Comparison of wireless techniques applied to environmental sensor monitoring." SINTEF Report (2017).

[4] Raza, Usman, Parag Kulkarni, and Mahesh Sooriyabandara. "Low power wide area networks: An overview." IEEE Communications Surveys & Tutorials 19.2 (2017): 855-873.

[5] "MulteFire™ lights up the path for universal wireless service." https://www.multefire.org/wp-content/uploads/2016/10/72-multefire-lights-up-the-path-for-universal-wireless-service.pdf. Accessed 4 Dec. 2018.

[6] Comparing IoT Technologies at a Glance | Wi-SUN Alliance." https://www.wi-sun.org/news/comp-iot-tech/. Accessed 4 Dec. 2018.

[7] Long Range Wireless IoT | 2018 Guide to LoRa and Other LPWAN …." https://www.postscapes.com/long-range-wireless-iot-protocol-lora/. Accessed 4 Dec. 2018.

[8] "Narrowband IoT - Wikipedia." https://en.wikipedia.org/wiki/Narrowband_IoT. 4.12.2018.

[9] "Cellular IoT – Part 9 – 50.000 devices per cell – WirelessMoves." 18 Sep. 2016, https://blog.wirelessmoves.com/2016/09/cellular-iot-part-9-50-000-devices.html. 4 Dec. 2018.

[10] Naik, Nitin. "LPWAN technologies for IoT systems: choice between ultra narrow band and spread spectrum." 2018 IEEE International Systems Engineering Symposium (ISSE). IEEE, 2018.

[11] Mikhaylov, Konstantin, Juha Petaejaejaervi, and Tuomo Haenninen. "Analysis of capacity and scalability of the LoRa low power wide area network technology." European Wireless 2016; 22th European Wireless Conference; Proceedings of. VDE, 2016.

[12] Chen, Min, et al. "Narrow band internet of things." IEEE Access5 (2017): 20557-20577.

[13] Talwar, Shilpa, et al. "Enabling technologies and architectures for 5G wireless." Microwave

[14] Muhammad A. Iqbal, Oladiran G.Olaleye and Magdy A. Bayoumi,”A Review on Internet of Things (Iot): Security and Privacy Requirements and the Solution Approaches”. Global Journal of Computer Science and Technology: ENetwork, Web & Security - 2016.

[15] Kejun Chen, Shuai Zhang, Zhikun Li, Yi Zhang, Qingxu Deng, Sandip Ray and Yier Jin,”Internet-of-Things Security and Vulnerabilities: Taxonomy, Challenges, and Practice”. Journal of Hardware and Systems Security - 10 May 2018.

[16] S. M. Riazul Islam, Daehan Kwak, Md. Humaun Kabir, Mahmud Hossain And Kyung-Sup Kwak,”The Internet of Things for Health Care: A Comprehensive Survey”. IEEE Access - June 1, 2015.

[17] Rashmi Sharan Sinha, Yiqiao Wei and Seung-Hoon Hwang,”A survey on LPWA technology: LoRa and NB-IoT”. Korean Institute of Communication and Information Sciences. March 14, 2017.

[18] Maria Rita Palattella, Mischa Dohler, Alfredo Grieco, Gianluca Rizzo, Johan Torsner, Thomas Engel, and Latif Ladid.” Internet of Things in the 5G Era: Enablers, Architecture, and Business Models”. IEEE Journal on Selected Areas in Communications, Vol. 34, No. 3, March 2016.

[19] The Future of Connectivity in IoT Deployments, Deloite Report.
[20] Richard Harada & Edgar Sotter,"Automated Monitoring for Inspections and Condition-based Maintenance- Part III: Industrial IoT Networks".
[21] Dong-gil Kim, Seawook Park, Kyungmin Kang and Dongik Lee," A Deterministic Wireless Network For Feedback Control Based On Ieee 802.15.4", School of Electrical Engineering and Computer Science, Kyungpook National University 1370 Sankyug-dong, Buk-gu, Daegu, 702-701, Korea - April 21, 2016.
[22] Kay Romer and Friedemann Mattern, "The Design Space of Wireless Sensor Networks". IEEE Wireless Communications • December 2004.
[23] C.Schmittner, Z. Ma, E. Schoitsch, T. Gruber, „A case study of FMVEA and CHASSIS as safety and security co-analysis method for automotive cyber-physical systems"
[24] Oliver Höftberger. Knowledge-Based Dynamic Reconfiguration for Embedded Real-Time Systems. PhD thesis, TU Wien, Institute of Computer Engineering, Wien, 2015.
[25] D. Ratasich, O. Höftberger, H. Isakovic, M. Shafique, and R. Grosu. A Self-Healing Framework for Building Resilient Cyber-Physical Systems. In 2017 IEEE 20th International Symposium on Real-Time Distributed Computing (ISORC), pages 133-140, May 2017.
[26] D. Ratasich, T. Preindl, K. Selyunin, and R. Grosu. Self-healing by property-guided structural adaptation. In 2018 IEEE Industrial Cyber-Physical Systems (ICPS), pages 199-205, May 2018.
[27] D. Ratasich, F. Khalid, F. Geissler, R. Grosu, M. Shafique, and E. Bartocci. A Roadmap Towards Resilient Internet of Things for Cyber-Physical Systems. IEEE Access, pages 1-24, Jan 2019.