Title:	*Final evaluation and validation report*
Authors:	*Miguel Angel Anton Gonzalez (TECNALIA), Emanuela Capotosto (ISRAA), Franck Chauvel (SINTEF), Jon Colado García (INDRA), Jacek Dominiak (Beawre), Nicolas Ferry (CNRS), Franck Fleurey (TellU), Anne Gallon (EVIDIAN), Rocío Gómez Robledo (INDRA), Modris Greitans (EDI), Uģis Grīnbergs (BOSC), Eider Iturbe Zamalloa (TECNALIA), Sergio Jiménez Gómez (INDRA), Vinh Hoa La (MONTIMAGE), Stéphane Lavirotte (CNRS), Saturnino Martinez Melchor (TECNALIA), Ruben Mulero (TECNALIA), Luong Nguyen (MONTIMAGE), Sarah Noye (TECNALIA), Alexander Palm (UDE), Francisco Parrilla Ayuso (INDRA), Angel Rego Fernandez (TECNALIA), Erkuden Rios Velasco (TECNALIA), Gérald Rocher (CNRS), Arnor Solberg (TellU), Jean-Yves Tigli (CNRS) and Oscar Zanutto (ISRAA).*
Editor:	*Tecnalia*
Reviewers:	*Nicolas Ferry (SINTEF), Andreas Metzger (UDE) and Hui Song (SINTEF).*
Identifier:	*D1.5*
Nature:	Deliverable
Date:	*08 April 2021*
Status:	*Draft*
Diss. level:	*Report: Public*

The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement n° 780351 (ENACT).

Executive Summary

Deliverable D1.5 describes the final validation and verification of the use cases and provides a summary of the general evaluation of the enablers developed in the ENACT project (at month 36).

The Use Cases environment comprises all the IoT infrastructures that are needed to perform Development and Operation of the Use Cases, including IoT and edge devices, the integration platforms and the cloud services. In this deliverable, the implementations developed during the final period of the project are tested against the use cases requirements and KPIs identified in the D1.1 to validate that the ENACT tools can be integrated in different kinds of environments, in a generic manner, using the infrastructure previously described in the D1.4. In addition, an overall evaluation of each enabler developed during this project is provided.

The three Use Cases implemented in ENACT are:

- **Intelligent Transport System (ITS):** The aim is to assess the feasibility of IoT services in the domain of train integrity control, in particular for the maintenance and logistics of the rolling stock and the on-track equipment.
- **Digital Health:** The aim is to provide medical device integration and data transport capabilities to external services such as an alarm centre or an electronic patient journal for stakeholders such as patients, doctors etc., exploiting the potential in IoT, edge and cloud services, adding more devices, actuation, distributed processing and better exploitation of collected data.
- **Smart Building:** The aim is to generate Smart Energy Efficiency applications and Smart Elderly Care applications, which will make use of sensors, actuators and services, in order to ensure the safety of the facilities to perform energy efficiency measures and also to support the care-takers in assessing the wellbeing of users.

The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement n° 780351 (ENACT).

Members of the ENACT consortium:

SINTEF AS	Norway
BEAWRE DIGITAL SL	Spain
EVIDIAN SA	France
INDRA Sistemas SA	Spain
Fundacion Tecnalia Research & Innovation	Spain
TellU IOT AS	Norway
Centre National de la Recherche Scientifique	France
Universitaet Duisburg-Essen	Germany
MONTIMAGE EURL	France
Istituto per Servizi di Ricovero e Assistenza agli Anziani	Italy
Baltic Open Solution Center	Latvia
Elektronikas un Datorzinatnu Instituts	Latvia

Revision history

Date	Version	Author	Comments
08.01.2021	0.1	Miguel Angel Anton	Table of Contents
10.01.2021	0.2	Sarah Noyé	Introduction, Method and Smart Building use case validation
26.02.2021	0.3	Sergio Jiménez Gómez	ITS case validation
26.03.2021	0.4	All partners.	Completion of use case validation and verification section; Overall Evaluation of the enablers; Conclusion and Annexes. Draft to be reviewed.
08.04.2021	0.5	Sergio Jimenez, Arnor Solberg, Miguel Angel Anton	Corrections after internal review.

Contents

Table of Tables

Table of Figures

Abbreviations and Definitions

Term	Definition
ACM	Actuation Conflict Manager
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
ARM	Advanced RISC Machine
BDA	Behavioural Drift Analysis
BLE	Bluetooth Low Energy
CTC	Centralized Traffic Control
DevOps	Development and Operations
eGW	Enhanced Gateway
EPJ	Electronic Patient Journals
ERTMS	European Rail Traffic Management System
GNSS	Global Navigation Satellite System
GRDP	General Registry of Data Protection
GSM-R	Global System for Mobile Communications – Railway
GUI	Graphical User Interface
GW	Gateway
HLA	High Level Architecture
HVAC	Heating, Ventilation and Air Conditioning
ID	Identification
IoT	Internet of Things
IRMS	Indra Rail Messaging Solution
ISO	International Organization for Standardization
ITS	Intelligent Transport Systems
JSON	JavaScript Object Notation
KP	Knowledge Processor
KPI	Key Performance Indicator
L&M	Logistics and Maintenance
LED	Light-Emitting Diode
MQTT	Message Queue Telemetry Transport
MW	Middleware
OPC-UA	OPC Unified Architecture
OPEX	OPerating EXpense

OSI	Open System Interconnection
OTI	On-board Train Integrity
PLC	Programmable Logic Controller
QoS	Quality of Service
RCA	Root Cause Analysis
RFID	Radio-Frequency Identification
S&P	Security and Privacy
SIB	Semantic Information Broker
SIS	Smart IoT Systems
SW	Software
T&S	Test and Simulation
TCP/IP	Transmission Control Protocol / Internet Protocol
TDS	Train Detection Systems
V&V	Validations and Verification
VM	Virtual Machine
VPN	Virtual Private Network
WSN	Wireless Sensor Network

1 Introduction and Objectives

This deliverable reports the work performed in the second phase of Task 1.3, led by Tecnalia, towards the final evaluation and validation of the ENACT tools for the three use cases of the ENACT project: Intelligent Transport System (ITS), Digital Health, and Smart Building. Indra and Tellu are responsible of assessing that the development and integration of the ITS and eHealth demonstrators respectively with the ENACT tools fulfilling the requirements stated in the D1.1. Tecnalia is responsible for the Smart Building demonstrator as a validation and verification platform. BOSC and EDI contribute to the validation and verification tasks of the ITS case study and facilitate the rail test infrastructure and demonstrations. ISRAA provide support for the Smart Building Use Case into the validation and verification tasks. In addition, all partners of the ENACT project contribute to the overall evaluation of the final version of the enablers developed in the project.

The Intelligent Transport System, the Digital Health, and the Smart Building use cases serve as evaluation platforms for the ENACT enablers. Each use case represents a different set of requirements from a different industry domain and different business model, with different needs regarding DevOps and trustworthiness. The three use cases also represent three different but typical DevOps scenarios of smart IoT systems: The Smart Building case shows the proof-of-concept development of applications in a testing environment: Kubik; The ITS use case represents the system integration work as the last step before production, where INDRA is the system integrator and EDI and BOSC are their hardware and software sensor suppliers; Finally, Tellu's Remote Patient Monitoring service is already in production, with hundreds of gateways used by real patients, and they need DevOps for the agile and continuous post-production evolution. This diversity of condition ensures that the ENACT tools can be transferred between different domains and different types of DevOps practices.

The objective of this evaluation is to test whether and how the ENACT tools enable and improve the DevOps practice for the development teams behind the three use cases, and how they improve the trustworthiness of the products. The results of the ENACT project are a set of development tools (i.e., the ENACT DevOps enablers), and therefore we do not expect the use case providers to directly integrate the ENACT tools into their vertical products. Instead, they use these tools to facilitate their development and operation activities behind the products. To achieve this, some of the tools are used directly for some particular tasks, some tools were integrated into the development environment, and some tools were integrated into the management systems that the use case providers developed to assist their main products. In these ways, the tools were gradually utilized by the use case providers to implement the three pilot systems. The final status of the three developed systems were reported in D1.4, together with which ENACT tools they used to develop the system, and how the tools were integrated into their development environment or management systems.

In order to evaluate the extent to which the tools enable and improve the DevOps practice, we defined a set of KPIs (aka, Test Objectives) based on challenges faced by the developers of the enablers, and the concrete test cases to check these KPIs, as documented in D1.1. In D1.3, we described the initial evaluation on how the use of ENACT tools helps fulfil these KPIs during the first part of the project, and in this deliverable, we report the final status of these KPIs after the second period of the project.

In Section 3, the use case validation and verification are presented, addressing the assessment of the finished tools with respect to the requirements and KPIs for each use case based on the validation and verification process introduced in D1.1.
Two type of tests are used to assess the KPIs and requirements:

- **ENACT tools KPIs:** Specific test to measure the deployment status of the last version of the different enablers in the use case. These efforts focus on improving the integration compatibility to cover a wider range of tools in a safe and secure manner.
- **ENACT tools validation and verification tests:** For each DevOps scenario defined in D1.1, the description and results of the testing and verification of associated enablers are shown. The

ENACT enablers have been tested in different DevOps scenarios and also each DevOps scenario comprises the testing and verification of several enablers.

- **ENACT Business KPIs:** In each use case, this concrete and technical evaluation is followed by a higher-level business evaluation, which describes how the development practice is improved by using the ENACT tools as a whole, and this improvement means to the use case providers in terms of higher productivity, increased revenue or reduced operation cost. This business evaluation is conducted based on a set of business KPIs.

In Section 4, an overall evaluation of each of the enablers after validation in the use cases is included as a conclusion to the ENACT project. This evaluation is intended to be a general conclusion of the status of the enabler after it has been tested in more than one use case. Two aspects are considered for each enables of the project: 1) The final evaluation summary of the enabler status, 2) the main conclusions and recommendation for future development and implementation of the enablers.

In section 5, the general conclusions of the final evaluation and validation of the ENACT enablers in the use cases are presented.

2 Use Case Validation and Verification

This section presents the validation and verification results for the different ENACT enabler in their corresponding use cases. The validation tests performed on the three use cases of the project are presented successively: ITS, Digital Health and Smart Building. Tests and KPIs are used to demonstrate the ENACT enablers have delivered the objectives set in the project.

The three use cases were developed independently to better assess the ability of the enablers to be generalised to different cases, but the methodology for the validation and verification is common so as to be able to compare test results between use cases. Different system tests scenarios are used to integrate and validate the enablers.

The tests have been designed to show how the ENACT enablers were improving the development and operation (DevOps) in the different use cases, including trustworthiness, without disturbing its business functionalities. The integration between the ENACT tools and the use cases is accomplished in a non-intrusive manner. The improvements that the ENACT tools add to the use case is tackled through the procedures, constrains, and inputs.

The methodology to obtain the KPIs focused on measuring how the ENACT enablers influence the DevOps of the use cases. It was divided into the following three stages according to the definition and analysis phases of the GQM [1] methodology:

- **STAGE 1**: Design of the roadmap for the evaluation of KPIs at the project level (D1.1 [1])
- **STAGE 2:** Performance of the KPI collection activities in a specific timeframe (M1-M22 in the ENACT project) (D1.3 [3])
- **STAGE 3:** Evaluate the KPI targets reached at the end of the project (D1.5)

The first and second stages were covered in the first period of the project and the stage 3 was performed in the third phase of the project. The impact of the ENACT enablers on the case study is presented in this section involving the following sequence of objectives:

- Evaluate the results obtained in order to draw the conclusions resulting from tests execution.
- Document the results achieved as well as draw the final set of conclusions.
- Present the results of ENACT KPIs project level evaluation.

The KPIs used for the validation of the enablers are divided in technical objectives following the ENACT tool groups:

- **TO1:** Support continuous delivery of trustworthy SIS.
- **TO2:** Support the agile operation of trustworthy SIS.
- **TO3:** Support continuous quality assurance strengthening trustworthiness of SIS.
- **TO4:** Leverage the capabilities of existing IoT platforms and fully exploit legacy, proprietary and off-the-shelf software components and devices.

The following sections summarize the verification and validation test performed in each use case based on the final use case implementation described in D1.4 [4]. The evaluation results are divided in two aspects:

1) the Use Case tests, which are the tests defined for the ENACT tools integration and functionality in the use case.
2) Use Case-ENACT tools tests, which are the use case specific testing features.

The reporting tables use the next colours to evaluate the level of progress:

- Green (Evaluation and validation is performed and KPI target is reached)
- Yellow (Some Evaluation and validation is performed and KPI is partly completed)
- Orange (Some Evaluation and validation is performed KPI is not fulfilled)
- Red (Evaluation and validation is not achieved)

In the first period, 50% of validation and verification was achieved. The second half has been completed in the second period. The final results of the project, including both the work carried out in the first and the second period are presented.

2.1 ITS Domain (Rail)

In the second period, the validation and verification of the ITS Use Case was focused on the ENACT DevOps tools and the added functionalities that the ITS infrastructure itself required to fulfil the ENACT tool requirements. The Test and Simulation (T&S) tool was integrated, but only from the messaging perspective, the T&S functionality itself had to be integrated with the Use Case itself. Moreover, the Behavioural Drift Analysis (BDA), Actuation Conflict Manager (ACM), GeneSIS, and Root Cause Analysis (RCA) tools required to be integrated. Moreover, the Security and Privacy (S&P) tool scenario has been built and validated. Further details in the section 2.1.1.In the second project period, the ITS Use Case validation and verification process was focused on the ENACT DevOps tools and the new ITS functionalities that are required to enable the ENACT tool requirements. The tools validated are:

- Test and Simulation (T&S) functionality itself (In the first period only the messaging was integrated).
- Behavioural Drift Analysis (BDA)
- Actuation Conflict Manager (ACM)
- GeneSIS
- Root Cause Analysis (RCA)
- Security and Privacy (S&P). For this tool,an specific scenario has been built and validated. Further details in the section 2.1.1.

It has been identified that these tools require not only Train Integrity and Logistics and Maintenance (L&M) functionalities data redefined in the D1.4 but also to retrieve information about the status of the infrastructure itself. A functionality, described in the deliverable D1.4 about the hardware, connectivity, and operational status of the gateways has been designed to this end and validated and verified in the second period of the project. Further details in the section 2.1.2.

A summary of the evaluation and validation performed applying the ITS Use Case is shown in Table 1, including both first and second period results. The summary is provided in the context of the identified KPIs related to the ITS Use Case (as stated in D1.1).

Code	KPI	Enabler involved	Status
TO1.1	Real Time Traffic Management Plan updates on On-Board Systems. Demonstrate the remote and continuous deployment of, at least, two cabins with OTI and a Plan update during the operation part. (i) Including at least 2 gateways and 2 IoT devices (ii) Including at least 1 cloud resource (iii) Including at least 2 deployments (iv) At least 1 software component is updated	GeneSIS	Completed. Deployment and update using rolling deployment techniques made in the ITS environment in 10 virtual gateways taking into consideration the operation state reported by these virtual gateways. In addition, deployment of FIWARE Orion Context Broker in the Cloud

TO1.2	SW development for the infrastructure deployed. Agile Software deployment on the CMWs Demonstrate a valid deployment with lack of human interaction in a reduced amount of time (limited by the device) increasing the efficiency in operation time and workload. Demonstrate the remote deployment of the CMWs: (i) Including at least 2 gateways and IoT devices, (ii) Including at least 1 cloud resource.	GeneSIS	Completed. Remote deployment made in the ITS environment in 10 virtual gateways taking into consideration (i) the operation state reported by these virtual gateways, and (ii) the release of a new software version (integration in Jenkins). The IoT edge devices are substituted by gateways as the available IoT edge devices cannot get Docker installed.
TO1.2	Elements simulation Simulation of at least 6 different devices	T&S	Completed. More than 60 devices simulated by the T&S tool.
TO4.1	Demonstrate the integration of the ITS SIS with the FiWARE (Orion Context Broker)	GeneSIS	Completed* The Use Case infrastructure is connected to FIWARE. Deployment of the FIWARE Orion Context Broker and Quantum Leap was demonstrated using GeneSIS and integrated with Grafana for displaying system status information. *GeneSIS is able to deploy all the FIWARE containers to make them functional. However, for the final demo the integration between the ITS infrastructure and FIWARE is made with an instance of FIWARE running in INDRA's cloud infrastructure.
TO2.1	Demonstrate the improvement of the monitoring thanks to contextual and effectiveness analysis at operational time.	BDA	Completed The model is created and deviation in the accelerometers can be predicted in advance. The addition of a simulator accelerates the generation of these models.
TO2.2	Identify and Solve actuations conflicts at design time.	ACM	Completed A model to prioritize the outputs that are generated in the cabin is created and deployed to be used
TO3.1	Attacks detection and isolation Demonstrate the detection and diagnoses different suspicious behaviours at the cloud and edge level respectively Isolate cybersecurity attacks for non-safety applications.	S&P	Completed. The connection between the On Board infrastructure and the S&P tool and the S&P Back End is made to detect the failures defined with rules.

TO3.2	Security not variable Simulate attacks and demonstrate the level of security keeps during runtime – i.e., protection against attacks works.	S&P	Completed. The security levels are not interfered by the S&P tool, the only interference is the user revoke process. This revoke process is planned in the scope.
TO3.3	Attacks historical Demonstrate access to an historical log of attacks in S&P Monitoring enabler	S&P	Completed. The S&P tool stores rules and send alerts that are stored about the attacks.
TO3.4	Orchestration Interface Demonstrate that the S&P Monitoring enabler is able to send alerts to the Orchestration and Continuous Deployment enabler.	BDA-GeneSIS	Remade It is been decided that the ITS infrastructure sends the alerts.
TO3.5	Failure Report Demonstrate that the enabler is able to analyses network data and provide information about, at least, one system source of failure.	RCA	Completed. The RCA tool is able to detect failures that were simulated in a previous stage to be detected during the operation.

Table 1: ITS - ENACT Tools KPIs

The KPIs described above are matched with the D1.1 test adaptations explained in the sections 2.1.1, 2.1.2, 2.1.3, and Appendix A.

2.1.1 Evaluation and Validation DevOps Scenarios

In addition to the evaluation and validation KPIs listed in the previous section, a set of DevOps scenarios related to the ITS Use Case are aggregated in a set of tests. These tests are intended to validate the various aspects encountered by the DevOps scenarios. The DevOps scenarios and the related set of tests were described in D1.1. The Table 2 summarizes the status of the tests designed for the ITS Use Case. The tests results are shown in the different sections of the document, the tests are further explained in the section 3 and the subsections 2.1.1.1, 2.1.1.2, 2.1.1.3, 2.1.1.4, 2.1.1.5, and 2.1.1.6. Moreover, an explanation of the relationship between the current performed tests and the ones stated in the D1.1 and D1.3 is made in the Appendix A.

Group of tests	Test	Description	ENACT tool	Test Update	Status
Things Data Monitoring (A.1.1.1.1)	Test 1.1.0.1 Monitoring On-Track data	Monitoring test	BDA	The test has been complete as the tool is able to record the behaviour of the sensors and generate a model about the proper behaviour of the sensors suing the accelerometers data.	Completed

Software update/Actuation (A.1.1.1.2)	Test 1.2.0.1 Wagon data update	Test for the wagon's identification data update	GeneSIS	The test has been changed, the parameters are not modified, instead, the security capabilities are modified based on the operational status reported by the gateways.	Completed
	Test 1.2.0.2 Orders prioritization.	Test to set the actuation over the system tasks prioritization	ACM	The test has been completed as the tool is able to create a prioritization logic for several outputs that occur in the train cabin.	Completed
	Test 1.2.0.3 Train stopped.	Test to realize the movement state of the train	GeneSIS	The gateways are able to provide its operational state to the tool and the tool is able to record this status and trigger a deployment accordingly.	Completed
Simulation and Testing tests (A.1.1.1.3)	Test 1.3.0.1 Single device Simulation testing.	Simulation of a single device test	T&S	The tool is able to record the behaviour of a single train to be replicated and, then, evaluate the scalability based on the most basic unit to measure, a single train composition.	Completed
	Test 1.3.0.2 Interaction between devices simulation testing.	Simulation of the Interactions between the devices	T&S	The tool is able to inject the data recorded to check how the gateway react to it and measure its performance.	Completed
	Test 1.3.0.3 Scaling procedure.	Simulation of the system scalability and problems that will be faced	T&S	The tool is able to reach the maximum capacity of the gateway based on tracking its performance to set its scalability limit factor.	Completed
Root Cause Analysis (0)	Test 1.4.0.1 Environmental distinction.	Test to distinguish the source of a communication issue	RCA	The tool is able to detect in operation a pre-trained failure.	Completed
Security Monitoring (A.1.1.1.5)	Test 1.5.0.1 Penetration testing.	Test of threads	S&P	The tool is able to detect predefined threats during the operation and act on them revoking the users that generate those failures.	Completed

Table 2: ITS - ENACT Tools Validation and Verification Test Progress

The tests were successfully completed during the second period of the project. The previous Table 2 shows the different KPIs defined in the D1.1 and D1.3.

Several integration tests have been accomplished in order to reach the DevOps tools objectives into the ITS Use Case. These integration tests require a specific extended infrastructure as an extension of the first project period one presented in the ENACT project middle review, the ITS Virtual Infrastructure.

The ITS Virtual Infrastructure is an extension of the gateway layer implemented in the first project period whose objective is providing an integration environment for the ENACT DevOps tools. It is formed by virtual replicas of the gateway that accomplish these functions:

- Routes the physical devices information to the storage and the representation mechanisms.
- Provides access to the FIWARE Cloud platform and the ENACT tools.
- Provides status information to the ENACT tools (RCA, GeneSIS and Testing and Simulation) to monitor data from several gateways.
- Provides an infrastructure to GeneSIS tool to deploy software on the CMW.

These actions are accomplished with three layers of gateways as it is shown in the Figure 1:

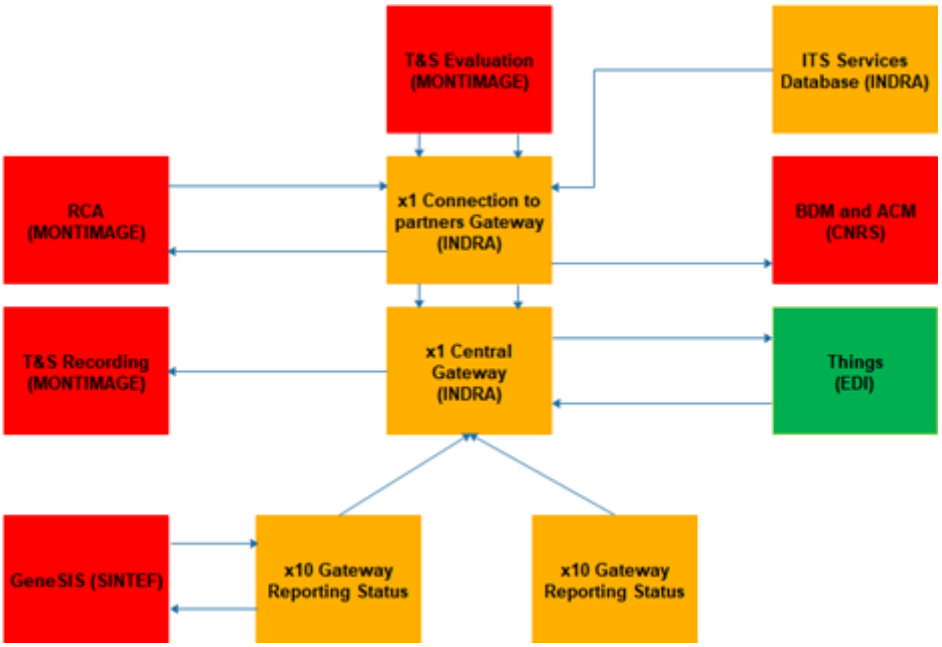

Figure 1: ITS Virtual Infrastructure

The three layers are explained:

- Connection to Partners Gateway: It provides access to the DevOps tools and to the FIWARE Cloud platform by AMQP.
- Central Gateway: This gateway is intended to replicate an On Board gateway, it is used to be connected with the Connection to Partners gateway. Moreover, it provides access to the Things remotely.
- Gateway Reporting Metadata: These gateways are used to simulate gateway status traffic and serves to GeneSIS to test Docker images deployments.

The methods to integrate the ENACT DevOps tools into this infrastructure is explained in the section 2.1.3.1. Moreover, all the information managed by the infrastructure is routed in an intelligent manner based on Routing topologies mechanism explained in the section 2.1.3.2.

It can be seen in the Figure 1 that different DevOps tools and ITS partners are connected. The main tools, in addition to the Things, connected to this infrastructure are listed:

- Orchestration and Continuous Deployment enabler with its associated GeneSIS tool.
- Test, Emulation and Simulation enabler with its associated T&S tool.
- Run-time Quality Assurance and Root Cause Analysis enabler with its associated RCA tool.
- Context Monitoring and Actuation Conflict Management enabler with its associated BDA and ACM tools.

These tools are explained in the next subsections:

2.1.1.1 Automatic Deployment - GeneSIS tool

The GeneSIS tool specific tests are explained in the section 3.1. As it is been explained in the previous deliverables, the GeneSIS tool is in charge of performing software deployments in the ITS Gateways for this Use Case enabling the ITS Gateway its security and safety capabilities. The security and safety capabilities are enabled into the ITS Gateway deploying different configuration into it, these configurations permit to the ITS Gateway to enable the connectivity between the ITS infrastructure clients (DevOps tools and ITS Use Case Things) and itself. The infrastructure used is shown in the Figure 1 as it is related with the GeneSIS tool and one of the x10 Gateway Reporting Status ITS Gateways group.

The Gateway Status Report is in charge of providing the systems status, that includes its capability to enable other entities (e.g., DevOps tools) to be connected to the ITS Gateways, further information in the section 2.1.3.3.

The ITS Gateway provides its operational metrics based on a configuration file. The following configurations files settings were included into Docker container to be deployed:

Container 1:

a. Services States Machine: 0 • Timer: 5 • Operational Status: 0
b. Attached repository: registry.gitlab.com/enact/its-app:0.5.0

This first configuration does not enable neither the connection to the ENACT DevOps tools nor the ITS partners (EDI).

Container 2:

a. Services States Machine: 0 • Timer: 5 • Operational Status: 1
b. Attached repository: registry.gitlab.com/enact/its-app:0.5.1

This second configuration does enable the ITS partners (EDI) connections.

Container 3:

a. Services States Machine: 0 • Timer: 5 • Operational Status: 2
b. Attached repository: registry.gitlab.com/enact/its-app:0.5.2

This second configuration does enable the ENACT DevOps tools and ITS partners (EDI) connections.

In the following Figures the processes that the GeneSIS tool accomplishes in the ITS Use Case are shown:

The first step for GeneSIS is to deploy the ITS-App, whose versions are available as Docker images in the ENACT Gitlab container repository [8]. In this scenario we deploy the ITS-App with some additional availability mechanisms. These mechanisms include watchdogs that monitor the application and automatically restart it when it fails. To do this, we provided GeneSIS with a health check script, which connects to the underlying Docker host and retrieves the status of the associated container. This script below shows this health check that the watchdog runs, every minute.

```
#!/usr/bin/bash

endpoint="$1"
host='192.168.1.162:2376'

container_name=$(echo "${endpoint}" | cut -d ':' -f 1);
container_status=$(curl -s "${host}/containers/${container_name}/json" | \
            jq .State.Status);
if [[ "${container_status}" =~ running ]];
then
    echo "Service is still running!"
    exit 0;
fi
echo "Service has failed!"
exit 1;
```

Figure 2: The health check script (Bash) used by the watchdog to detect failures of the ITS application. See the associated video

In this Bash script, the IP address of the Docker host is fixed to 192.168.0.1. This is the local address of the underlying host within the Docker network that GeneSIS automatically creates to connect the proxy container where the watchdogs are running to the running instances of the ITS app and the underlying docker host.

To showcase availability, we simulated a failure of the ITS-app by logging onto the gateway and stopping the container where the ITS-app runs. From there, we log onto the proxy container deployed along by GeneSIS where the watchdog is running and we observe that it detects the failure and automatically provisions a replacement, reducing the interruption to a minute, that is the time between two runs of the health check script. The following screenshot extracted from associated video shows the log file generated by the watchdog, where we can see that detected the failure of the component 'its-app-1' and provisioned a replacement.

```
root@747db011fab9:/enact# cat watchdog_its-app-1.log
Service is still running!
Tue Mar 16 11:37:01 UTC 2021 INFO: Heath check for 'its-app-1:5000' return
ed '0'
Tue Mar 16 11:37:01 UTC 2021 INFO: Endpoint 'its-app-1:5000' is back!
Endpoint already known.
Endpoint 'its-app-1:5000' is already active!
Service is still running!
Tue Mar 16 11:38:01 UTC 2021 INFO: Heath check for 'its-app-1:5000' return
ed '0'
Service is still running!
Tue Mar 16 11:39:02 UTC 2021 INFO: Heath check for 'its-app-1:5000' return
ed '0'
Service has failed!
Tue Mar 16 11:40:01 UTC 2021 INFO: Heath check for 'its-app-1:5000' return
ed '1'
Tue Mar 16 11:40:01 UTC 2021 INFO: Endpoint 'its-app-1:5000' has failed!
Configuration loaded from '/enact/docker.sh':
  - Host: '192.168.0.1:2376'
  - Image: 'registry.gitlab.com/enact/its-app:0.3.0'
  - Command: 'python app.py'
  - Network: 'GeneSIS-its-app'
Container 'its-app-1' stopped.
Container 'its-app-1' removed.
New container 'its-app-1' from image 'registry.gitlab.com/enact/its-app:0.
3.0' created!
ID: "a5f7227b75ca5b522c027b225fc0277017fd6e8d99b1660cce6e235e5cc9fd0c"
Container 'its-app-1' connected to 'GeneSIS-its-app'
Container 'its-app-1' started!
No backup endpoint for 'its-app-1:5000'.
```

Figure 3: Looking at the watchdog log file. We see that the failure of the ITS app was detected and automatically mitigated

Besides, we configured GeneSIS to minimise service interruptions during upgrades. Since this application cannot be replicated, upgrading to a new version implies to stop the application and provision the new version. To minimise the associated down time, which is in order of minutes because it requires pulling a new Docker image, GeneSIS pulls the image before to stop the old container. It first prepares, but does not start the new container. Only once it has pulled the new image and created the associated

container, GeneSIS stops the older version and starts the new one. The interruption thus boils down to the bootup time of the ITS application, that is in the order of seconds. This could only be achieved by leveraging the GeneSIS support for rolling deployment, which provide finer grained configuration options than what can be generally found in solutions such as Docker Swarm.

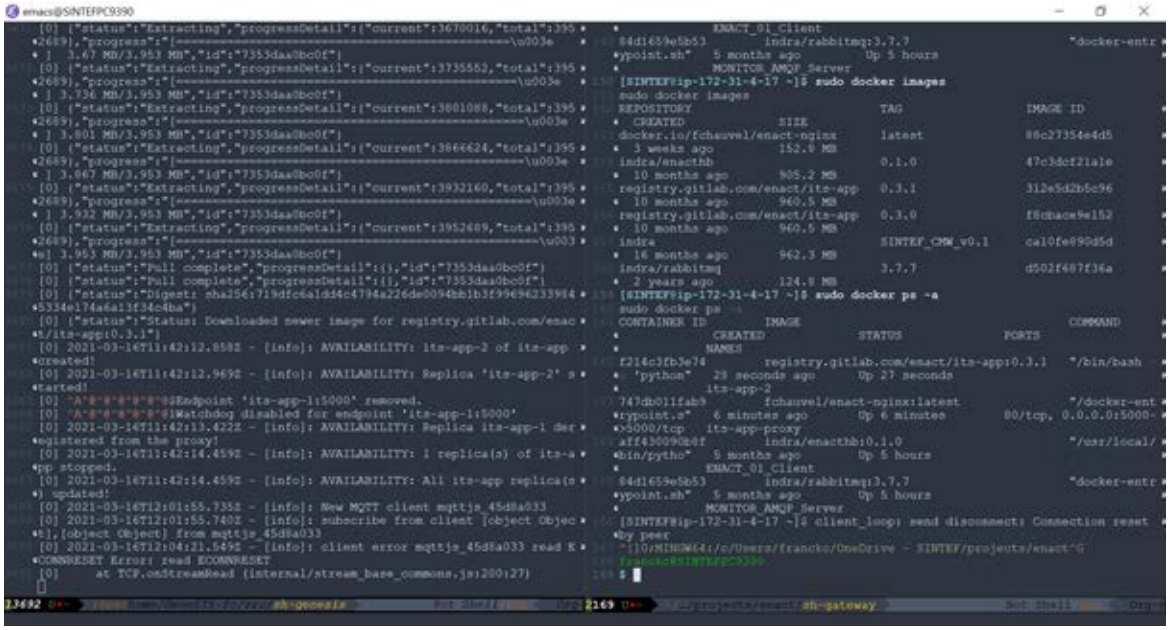

Figure 4: Traces of the upgrade of the ITS application. GeneSIS prepares the new version before its stops the older one

As we can see in the screenshot extracted from the associated video, GeneSIS (on the left panel) creates the new container before it stops the old one. On the right panel, we can see that a new container named "its-app-2", instantiated from the new version (v0.3.1), has been created.

Regarding the KPIs validation, the TO1.1 and TO1.2 are validated as the deployment is made correctly and in a secured manner in the number of machines that the TO1.1 KPI stated. The TO4.1 is validated as it was observed in the middle review; however, for the final demo the FIWARE capabilities deployed are made by INDRA. Finally, the TO3.4 is validated, but the entity that provided the alerts to GeneSIS is the ITS infrastructure itself. This decision is based on the addition of the Gateway Status and Operational Stage Provision functionality, explained in the section 2.1.3.3.

2.1.1.2 Testing and Simulation tool (T&S)

The T&S tool, further explained in the section 4.2, section shows the material obtained in the integration tests between the T&S tool and the ITS Use Case. It must be emphasized that the only ITS infrastructure elements used for these tests are the T&S tool, Central gateway, Connection to partners Gateway, x20 Gateway Reporting Status, and the Things.

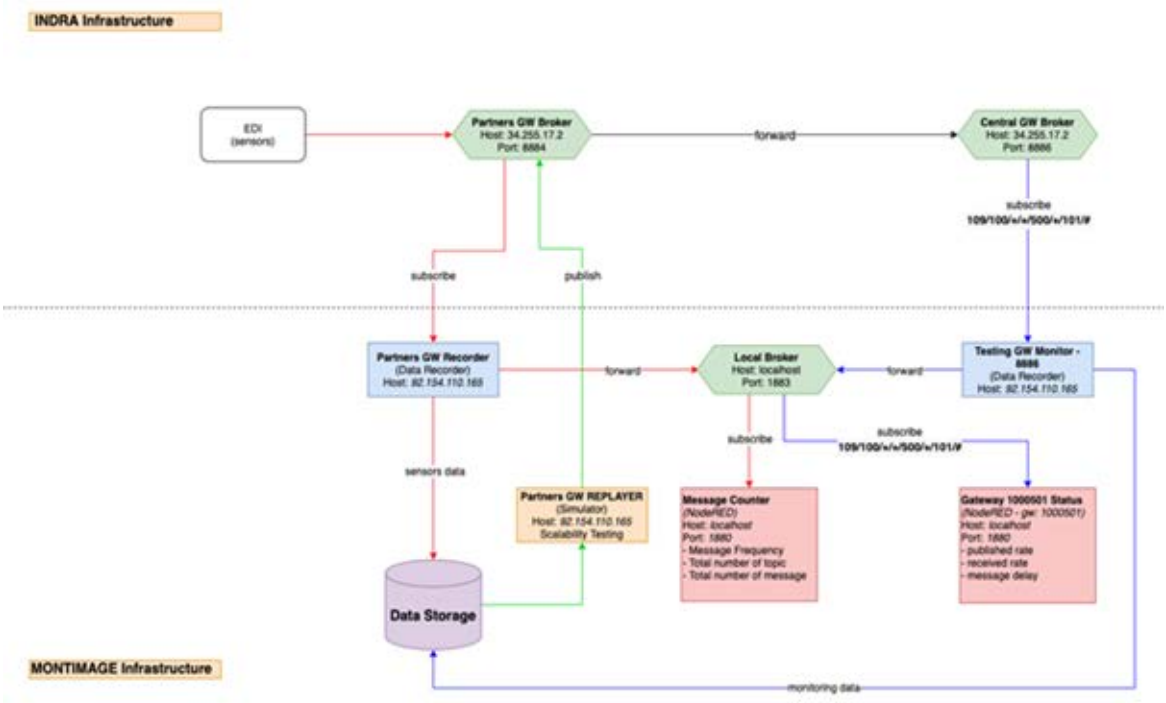

Figure 5: ITS - T&S Integration Arquitecture

As it can be seen in the Figure 5, the tool is integrated with the Central Gateway and the Connection to partners Gateway mentioned in the Figure 1. The architecture simulates a real train with the Central Gateway, which is the ITS Gateway to evaluate, and the Connection to partners Gateway, as a Cloud platform, that is used to get the Central Gateway status reports. This information is treated internally in the T&S tool to store the recorded data and to reinject it to the Central Gateway, while the gateway reports are received to check the Central Gateway status.

The tests are divided in two stages. The first one is made to record the normal behaviour of a single train as, at the end, the train is the atomic unit to be replicated (Central Gateway and Things). The second stage consist in reinjecting the recorded data to check if the gateway can deal with a specific number of trains.

The recording stage is shown in the following figures. The Figure 6 show the prerecording phase and the Figure 7 show the after recording stage.

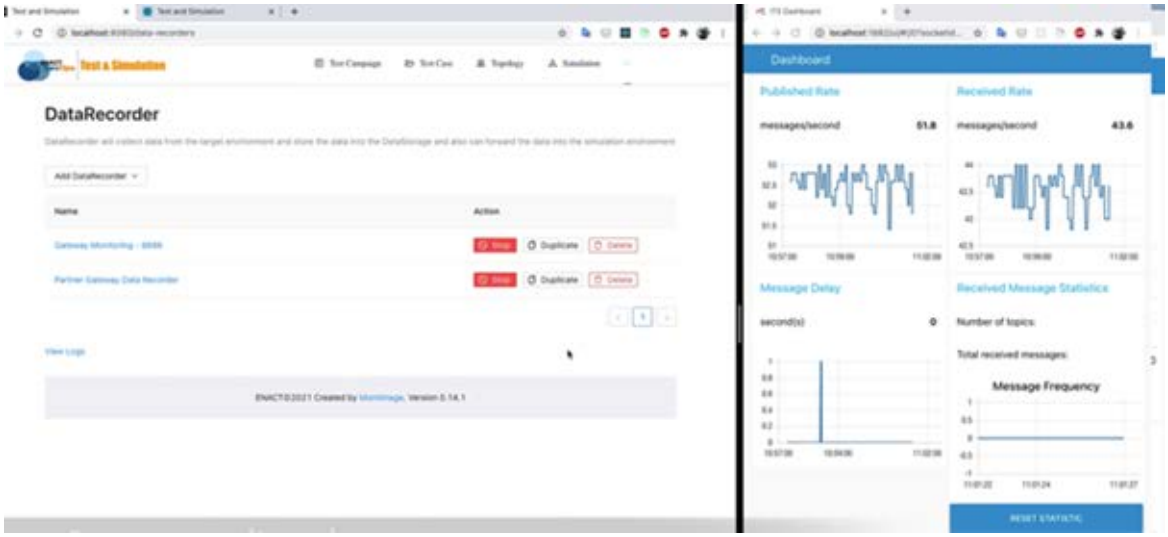

Figure 6: ITS - T&S Pre-record Stage

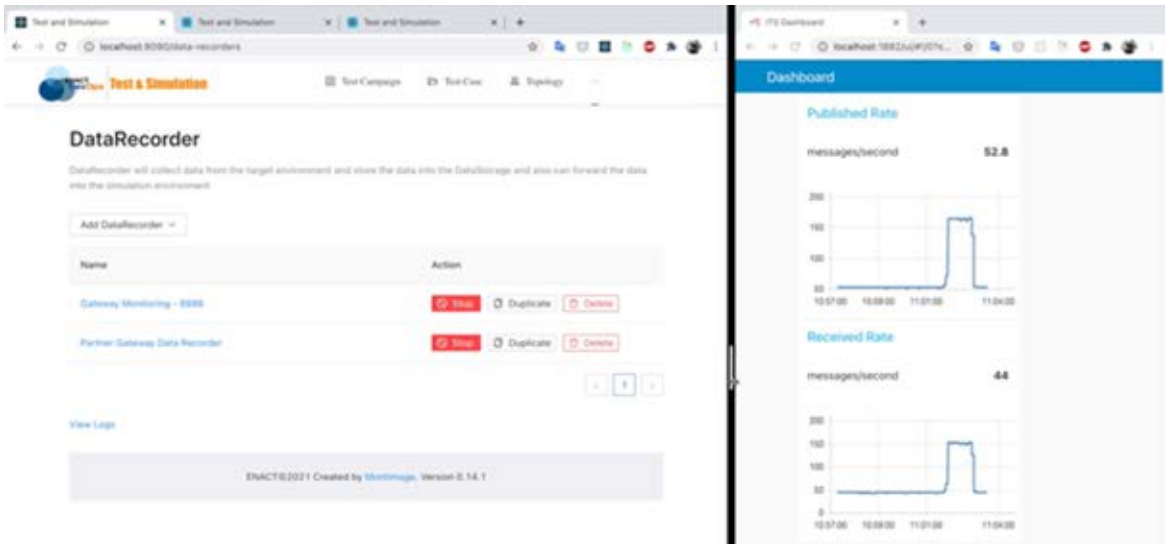

Figure 7: ITS - T&S After-record Stage

As it is observed, the traffic peak is recorded. This traffic is constant as the safety systems are required it. The Figure 7 show some traffic volume when the recoding is not active, it is the status of the gateway even when no traffic from the train is published.

The recoding information was made with 3 Things that have three sensors each and the gateway status of the Central Gateway itself; therefore, 13 devices connected per train. Several steps are given to scale these devices' traffic to be reinjected at this stage:

- Scale factor 1: 13 devices.
- Scale factor 5: 65 devices.
- Scale factor 10: 130 devices.
- Scale factor 20: 260 devices.
- Scale factor 50: 650 devices.

The Figures shows that for the steps 1,5,10, and 20, the messages are carried with no further issues. However, once the traffic is replicated 50 times the central gateway started to show delays and the outputs messages are not the same than the input messages as failure indicator. This means that the ITS Gateway is starting to queue the messages (Figure 12).

The information to get the Central Gateway state is given by the Gateway Status Report explained in the section 2.1.3.3 recorded by the T&S tool in the Connection to partners Gateway.

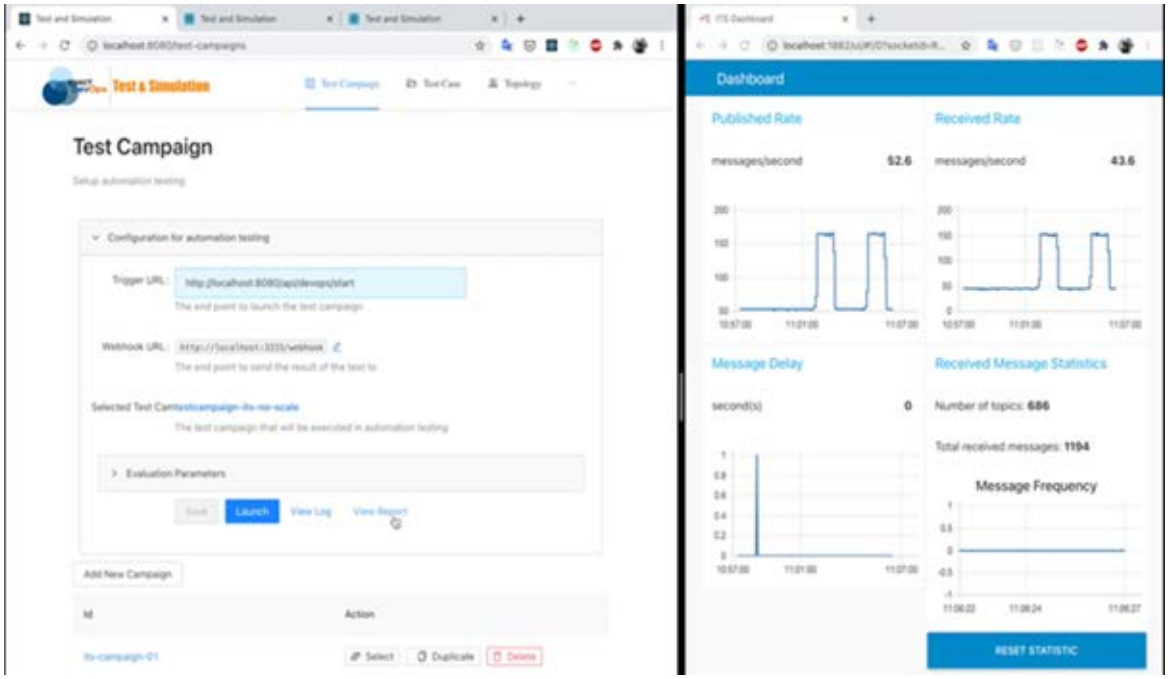

Figure 8: ITS - T&S Reinjection - Scale factor 1

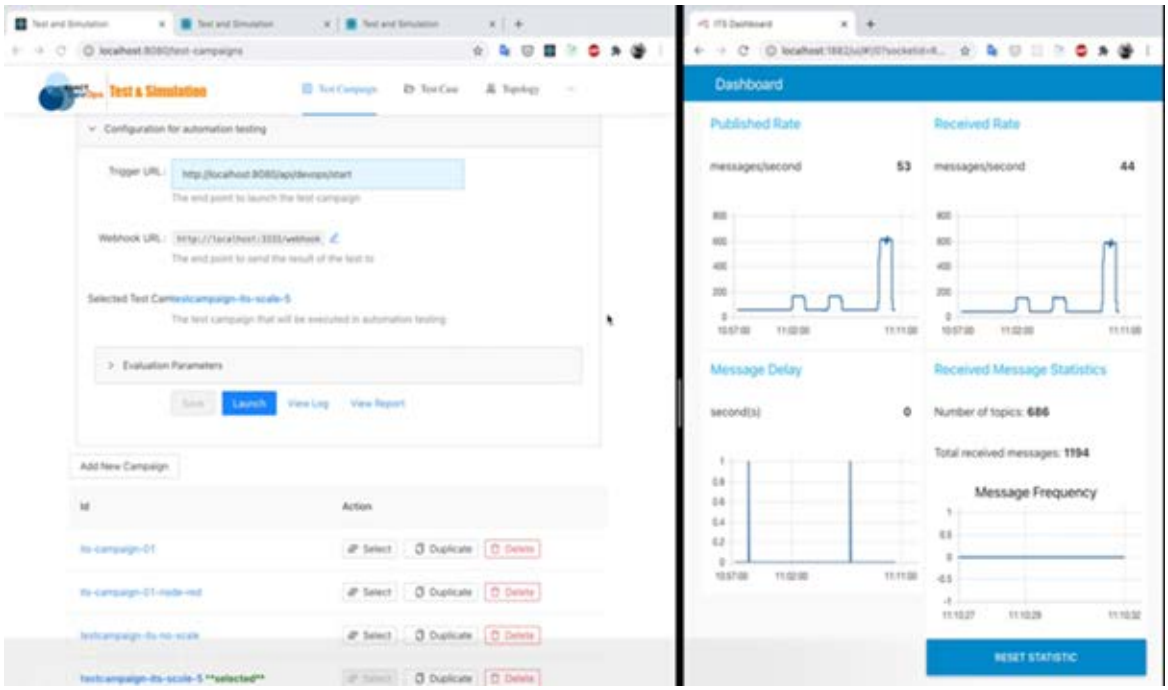

Figure 9: ITS - T&S Reinjection – Scale factor 5

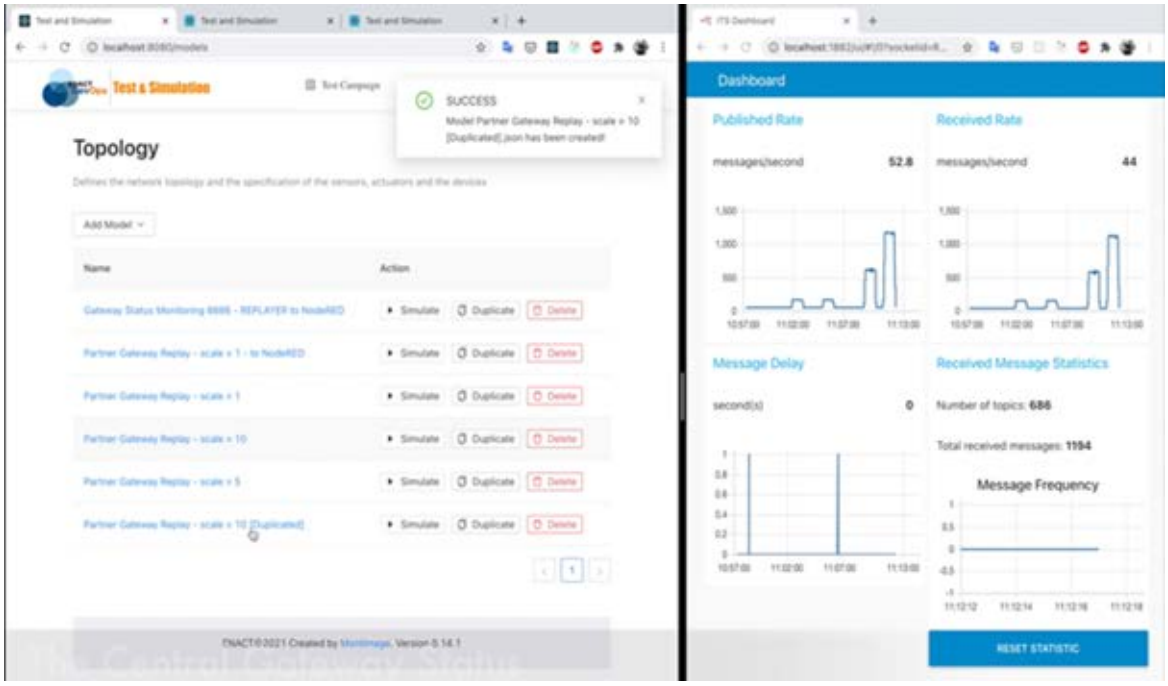

Figure 10: ITS - T&S Reinjection – Scale factor 10

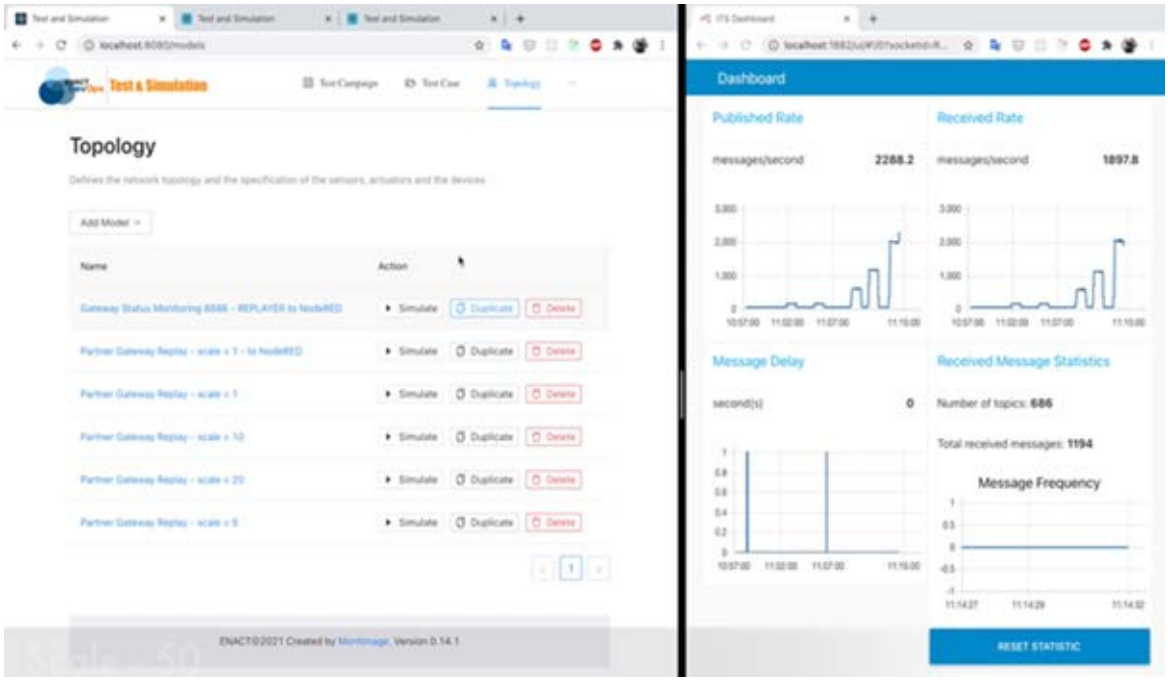

Figure 11: ITS – T&S Reinjection – Scale factor20

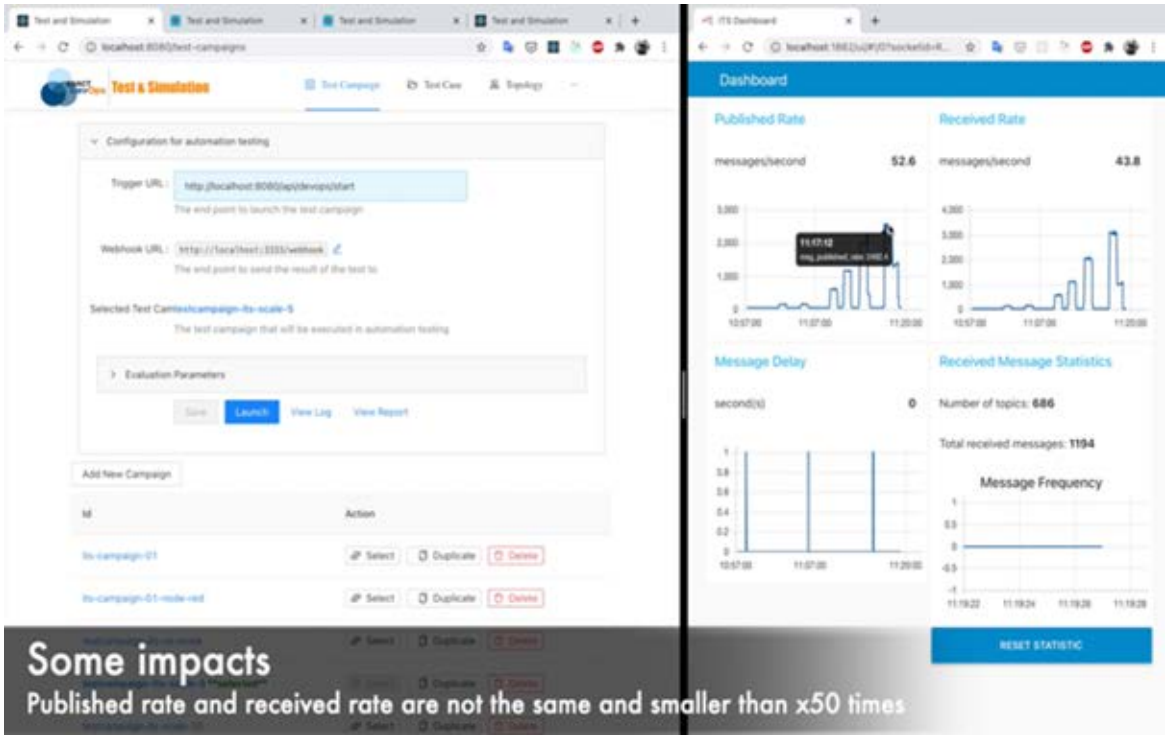

Figure 12: ITS – T&S Reinjection – Scale factor 50

As it is observed, the 20 trains step is carried by the ITS Gateway with no problem. It implies that more than 100 devices can be handle by the ITS Gateway ensuring the scalability criteria for the ITS Use Case. It can be concluded that the ITS Gateway is able to carry the 100 devices stated in the ENACT DoA and, then, the KPIs validated (TO1.2).

2.1.1.3 Root Cause Analysis tool (RCA)

The RCA tool, further explained in the section 3.4, analyses the infrastructure performance and helps to better understand the origin of the degradation of the overall performance. The tests consist in recording different infrastructure failure scenarios that may occur during its operation. These failures are forced to be recorded in a controlled manner; therefore, the RCA tool can learn them and detect their reoccurrence during the operation. It must be emphasized that the ITS infrastructure elements used for these tests are the RCA tool, Central gateway, Connection to partners Gateway, x20 Gateway Reporting Status, and the Things.

Figure 13 demonstrates the overall architecture of the tests, in which the two failures injected to the ITS system are as follows:

- The failure 1 consists in removing the Things users; therefore, they cannot publish business data. As a consequence, the number of messages per second should decrease and can be the failure indicator for the RCA tool.
- The failure 2 consists in removing the permissions to the central gateway to report the Things messages to the Connection to partners Gateway. Consequently, the number of messages should decrease and can be a failure indicator for the RCA tool.

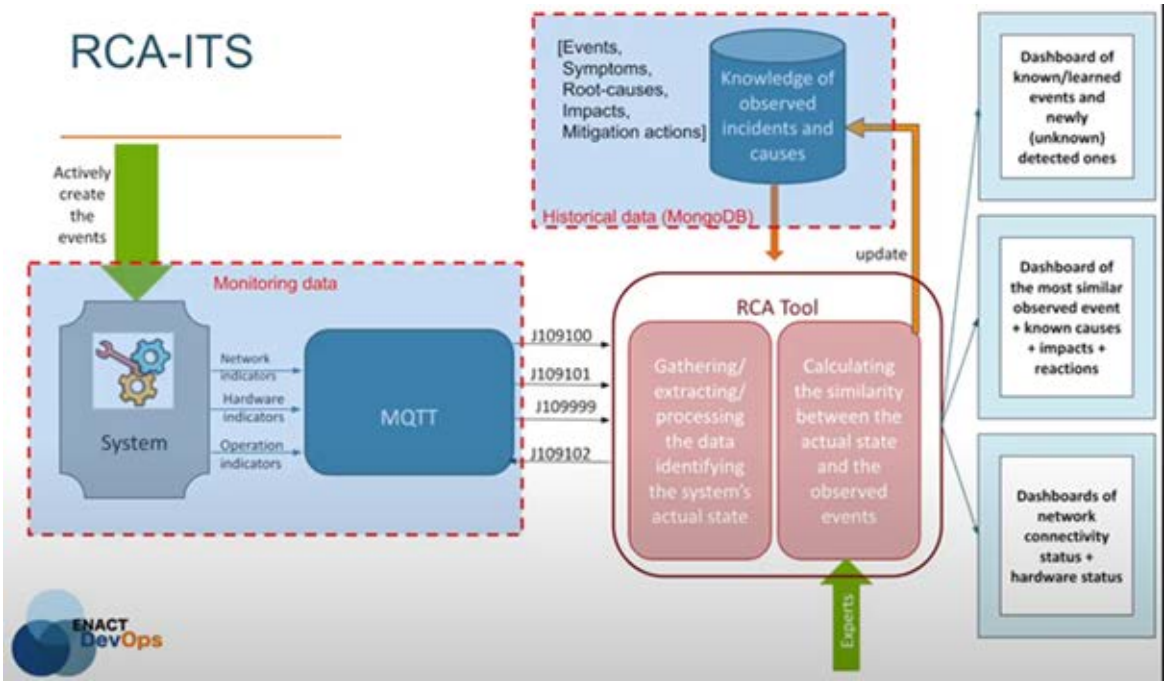

Figure 13: ITS – RCA Integration Architecture

As presented in the Figure 13, the monitoring data (i.e., J109100 - network/ J109101 - hardware/ J109999 - operation) is transferred from the ITS Virtual infrastructure to the RCA tool via MQTT connections. It will be further processed and analysed by the RCA. .The tests can be divided into two different stages: the recording/learning stage and the monitoring stage. The former is for building a historical database (Mongodb) of known normal and abnormal states (two aforementioned failures). Whilst, the latter consists of monitoring the system in real-time, analysing the current state by querying the historical data. This involves using the concept of Similarity Learning. That means the tool calculates the "similarity score" of a current state with the known normal and abnormal ones and to determine if the system is likely properly functioning or under failures/ errors.

The RCA tool consists of three different (group) of dashboards:

Dashboards of connectivity and hardware status: These dashboards (Figure 14, Figure 15) allow to visualise in a straightforward manner the monitoring data received from the MQTT broker and to observe simple changes/ incidents in the system.

Figure 14: ITS - RCA Connectivity Report

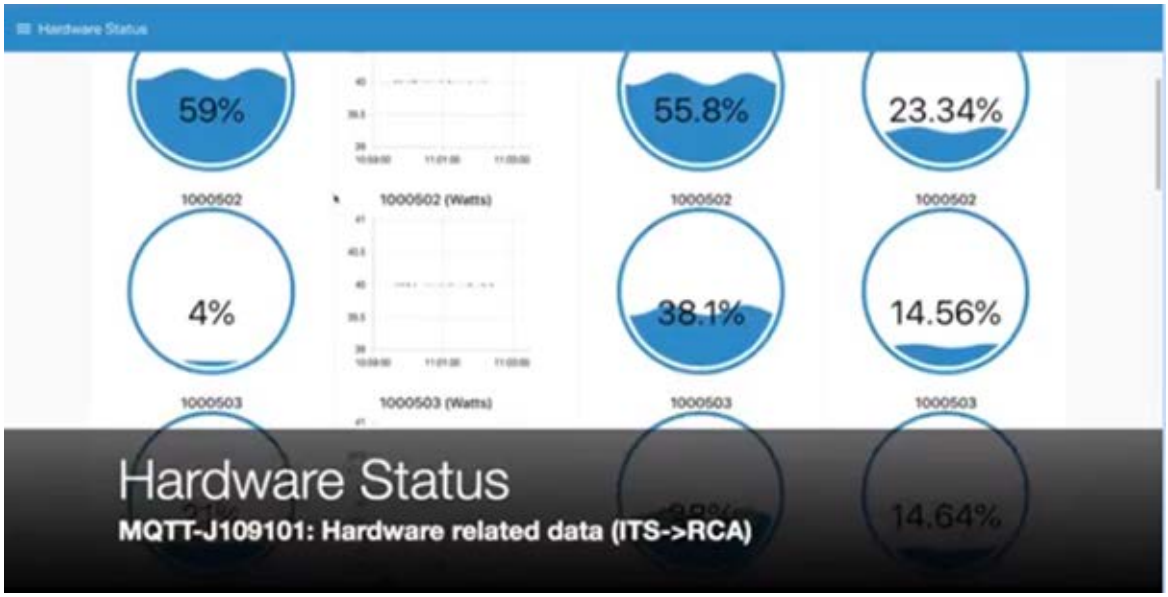

Figure 15: ITS – RCA Hardware Report

Dashboards of known/learned states (Figure 16) and newly detected (by third-party tool) incidents/ behaviours. A ranked list of the most similar known states in comparison with the new one. However, in the evaluation with ITS, no third-party tool was used.

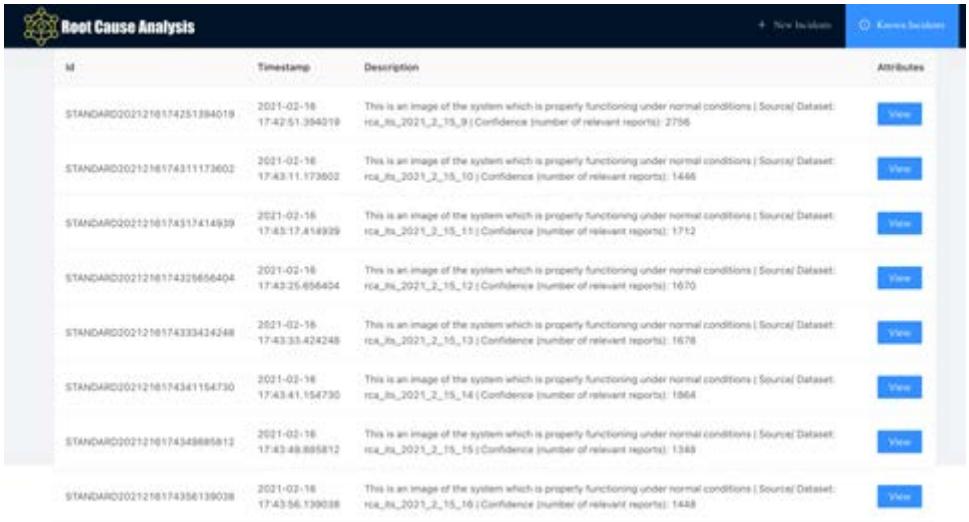

Figure 16: ITS - RCA – Properly functioning states learned

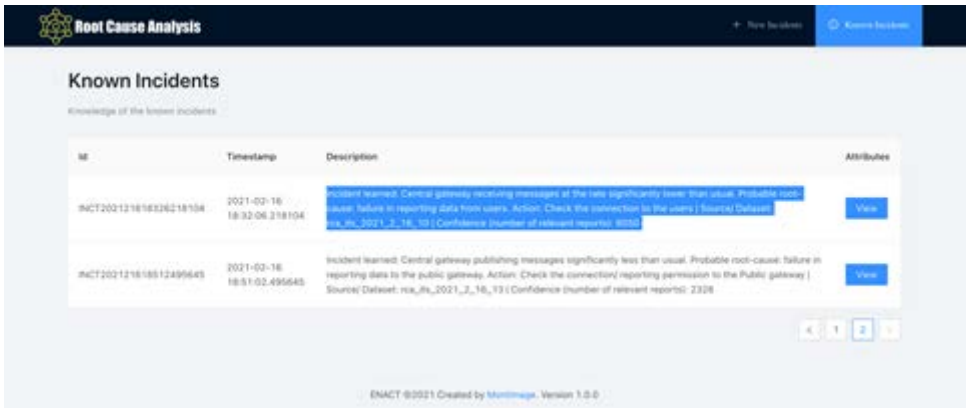

Figure 17: ITS - RCA Failures 1 learned

Figure 18: ITS – RCA Failure 2 learned

Dashboard of real-time analysis result visualises how the current state of the system is likely more similar to a normal or an abnormal state.

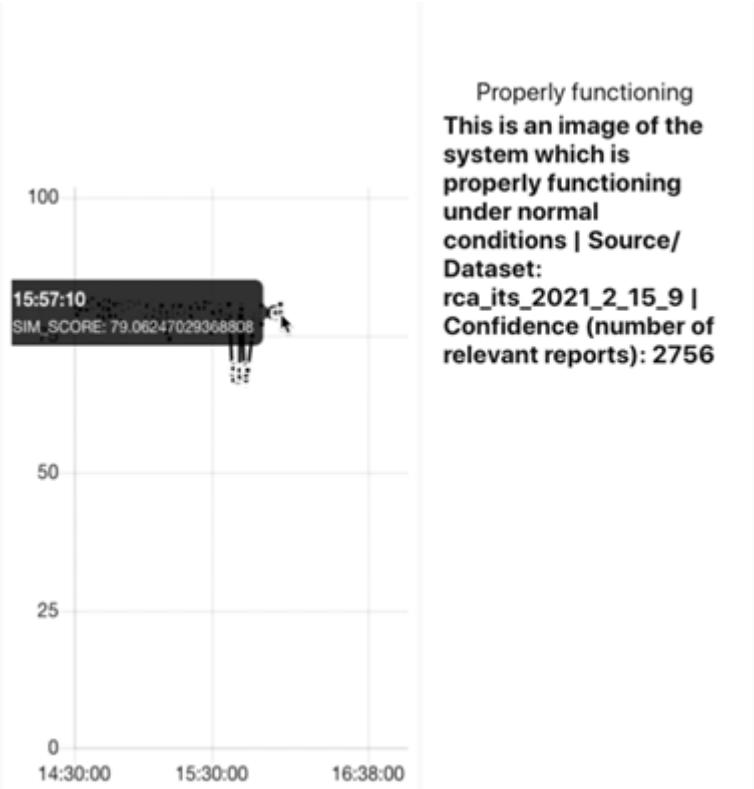

Figure 19: ITS - RCA - The current state is more similar to a normal one

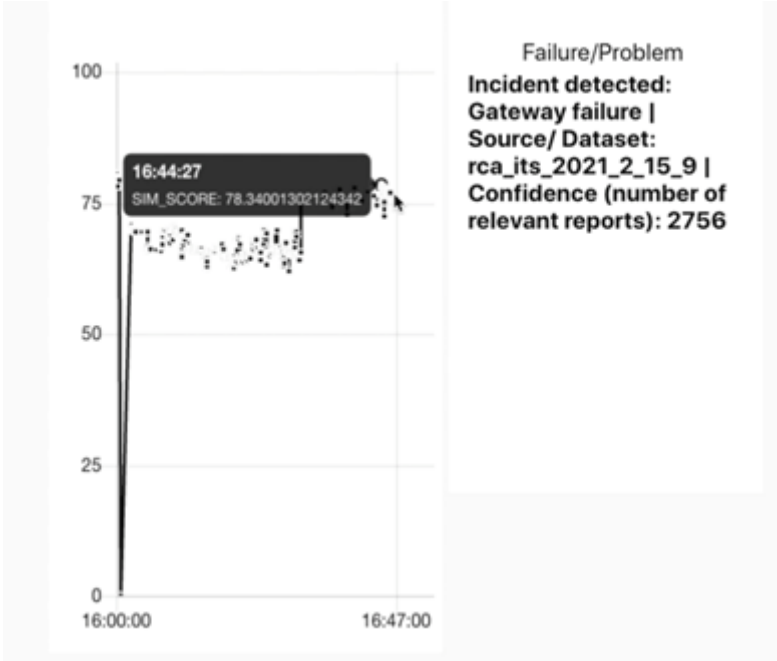

Figure 20: ITS - RCA - The current state is more similar to a learned failure

In summary, the RCA monitors the ITS system based on provided monitoring data as expected: The injected failures resulted in a low similarity score to the captured normal states. In contrast, once the failures were learned by the RCA, their reoccurrence triggered high score to the failures themselves, and low score to the properly functioning states. Concrete details will be presented in section 4.4.

As a result, regarding the KPIs validation, the TO3.5 has been validated as the RCA tool is able to analyse network data and determine the sources of two pre-trained failures. Indeed, RCA provides the way to systemize the observed incidents, symptoms, causes, impacts and mitigation actions in both active and passive manner, as well as relies on machine learning algorithms to identify the probable cause(s) of the detected anomalies based on the knowledge of similar observed ones.

2.1.1.4 Actuation Conflict Management tool (ACM)

The Actuation Conflict Management enabler is further explained in section 4.5. As presented in the previous deliverables, the Actuation Conflict Management enabler is in charge of detecting and solving actuation conflicts (actuation orders with conflicting goals). As part of this use case, conflicts may arise when applications emit information to the equipment used to interact with the driver. The conflicts management helps the rail environment to ensure, in non-safety applications, the proper prioritization of the orders to be all executed properly.

It should be noted that since it was obviously not possible to perform and test actuation conflicts on a real train and because the hardware infrastructure developed in the project was mainly focused on sensors with a limited number of actuators, thus hindering the validation of complex actuation conflict configurations, this module was tested with an advanced train simulator called OpenBVE. Nevertheless, the interactions between the train and the module were performed in an "as-if" scenario - i.e., in a configuration similar to the one that could have been performed on a real train. In addition to the OpenBVE simulator, physical actuators have been added to the overall evaluation environment. In particular, an alarm system was developed using an Arduino platform with two audio broadcast channels: an alarm and an audio message broadcast system. The application running on the Arduino was realized with ThingML[1].

Three applications have been developed with Node-RED that are applied to train control and communication between the train driver, the network controller and the passengers in the wagons.

- The first application is a dashboard for the train driver. The actions triggered via this dashboard are reflected in the train simulator used to drive it.
- A second simple application allows train passengers to request the opening or closing of doors, to trigger an alarm signal or to communicate with the train driver.
- The last application is the traffic controller (Figure 21). It provides access to the various train's sensors, allows the train to be stopped remotely in the event of an emergency, and to establish communication with the train driver.

[1] https://gitlab.com/enact/actuation_conflict_manager/-/tree/master/demos/thingml

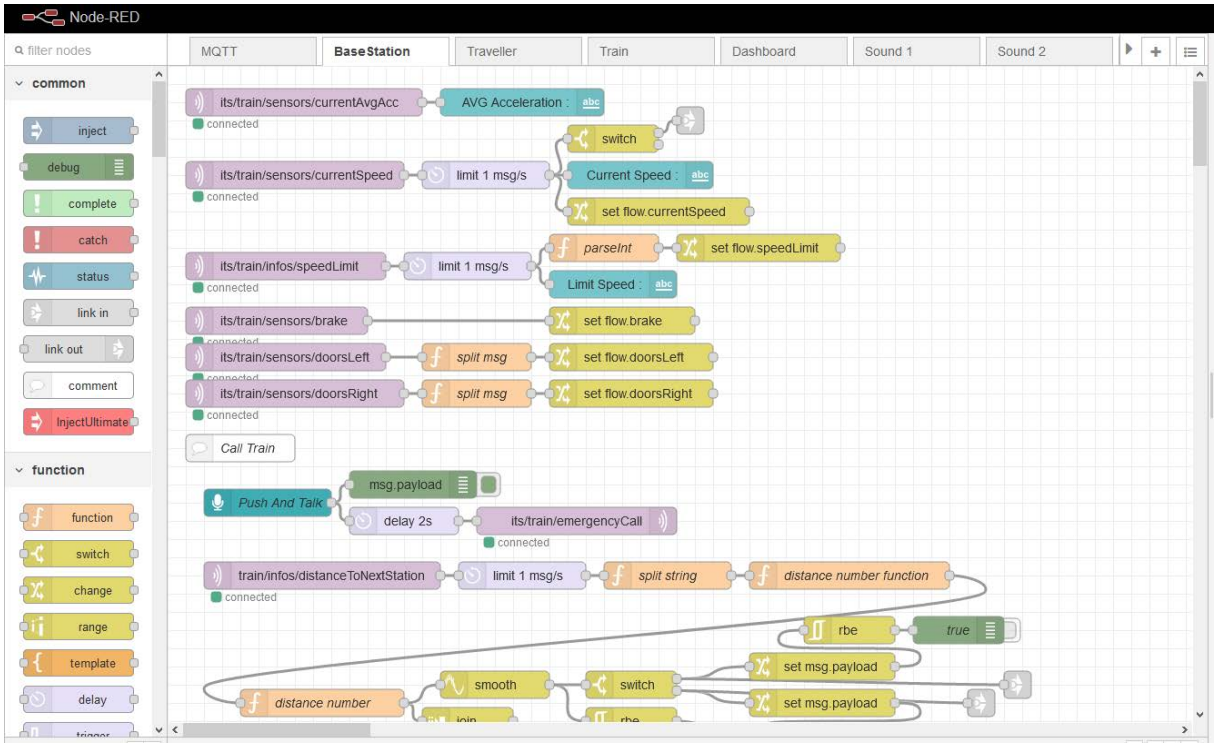

Figure 21: Actuation Conflict Management applications

These three applications were developed by different members of the DevOps team. The use of the three applications together showed the occurrence of situations where there was a conflict between the different actions performed by each of these applications (for instance, a passenger contacts the train driver when he is already in communication with the traffic management person and at the same time there is an emergency alarm). A manual search for conflicts was undertaken but quickly abandoned, as there was no guarantee of having found all of them. The use of the ACM enabler allowed the detection of all potential conflicts in the case study including not only the conflicts already identified by the manual search, but also found two others that had not.

The ACM enabler, through a GeneSIS deployment model and a model of the environment, builds a WIMAC model. This model is used to identify and solve the conflicts. In this case, the WIMAC model had 710 components divided into as many nodes and links (Figure 22).

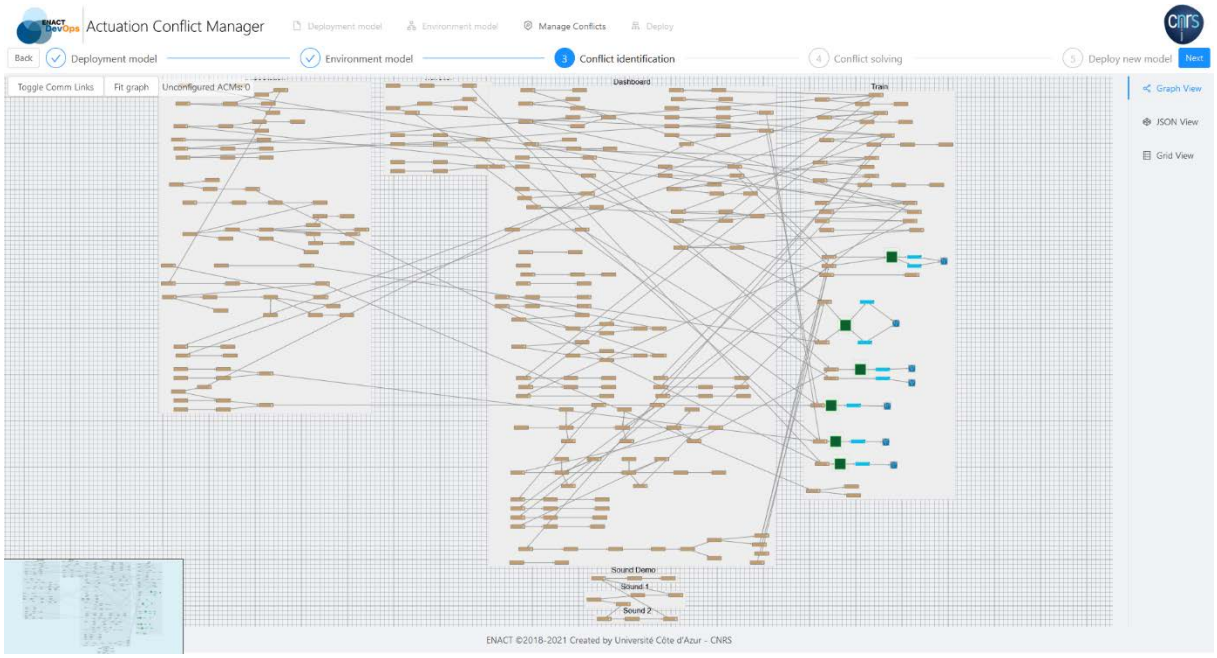

Figure 22: Actuation Conflict Management enabler on ITS application

A total of six conflicts were identified. One of these conflicts involved the ThingML application (introduced to manage an alarm device) and the resolution of this conflict led to the injection of an ACM component into the ThingML assembly, demonstrating the interest of a language independent detection and resolution tool. The other identified conflicts were on applications developed in Node-RED. The potential conflicts were all identified and resolved.

Two of the identified conflicts were indirect conflicts and the other four were direct conflicts. The two indirect conflicts were related to the speed instructions given to the train (acceleration, braking and emergency braking), the other one to an audio conflict in the driver's cab (audio message from a passenger, audio message from the network controller and alarm message broadcasted by the driver's failure to respect a speed instruction).

Regarding the KPIs validation, TO2.2 is validated as the detection and resolution of actuations conflicts is achieved using ACM tool at design time, before deploying the applications.

2.1.1.5 Behavioural Drift Analysis tool (BDA)

The Context and Behavioural Drift Analysis (BDA) enabler is further explained in section 4.6. The BDA was used in two contexts: with the ITS virtual infrastructure and with the OpenBVE train simulator (Figure 23). The later was used because it provided a means to create multiple situations where the train behaviour deviated from the expected behaviour, which would not have been possible with a real infrastructure.

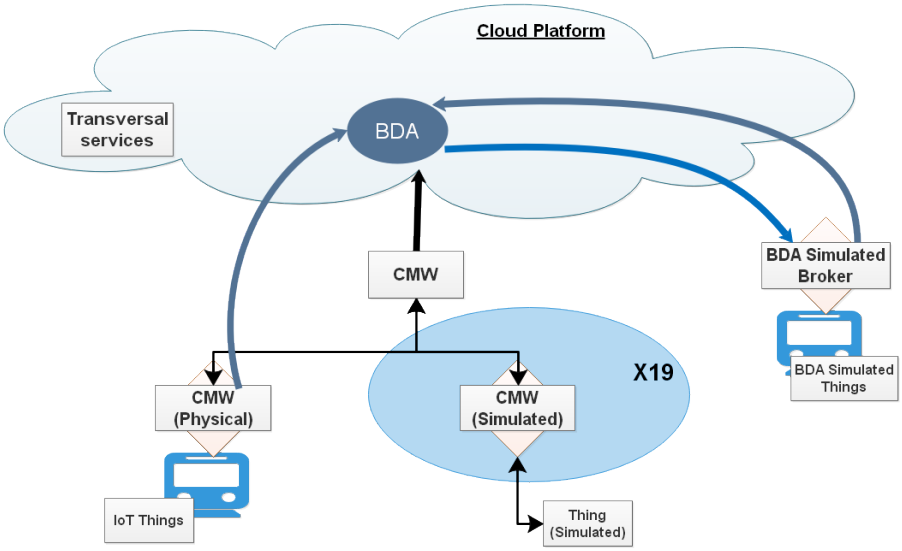

Figure 23: Behavioural Drift Analysis enabler and ITS infrastructure

In order to validate the integration of the tool with the ITS infrastructure (CWM), messages from the ITS virtual infrastructure provided by the sensors embedded on the EDI mini-train were used to feed the Behavioural Drift Analysis algorithm. A behavioural drift model was specified, providing information on the expected behaviour of the two trailing wagons of the train. This model (Figure 24) specifies the expected values for the accelerometers (nodes 6 and 7 accelerometers values displayed on Grafana on Figure 26) depending on different train speeds.

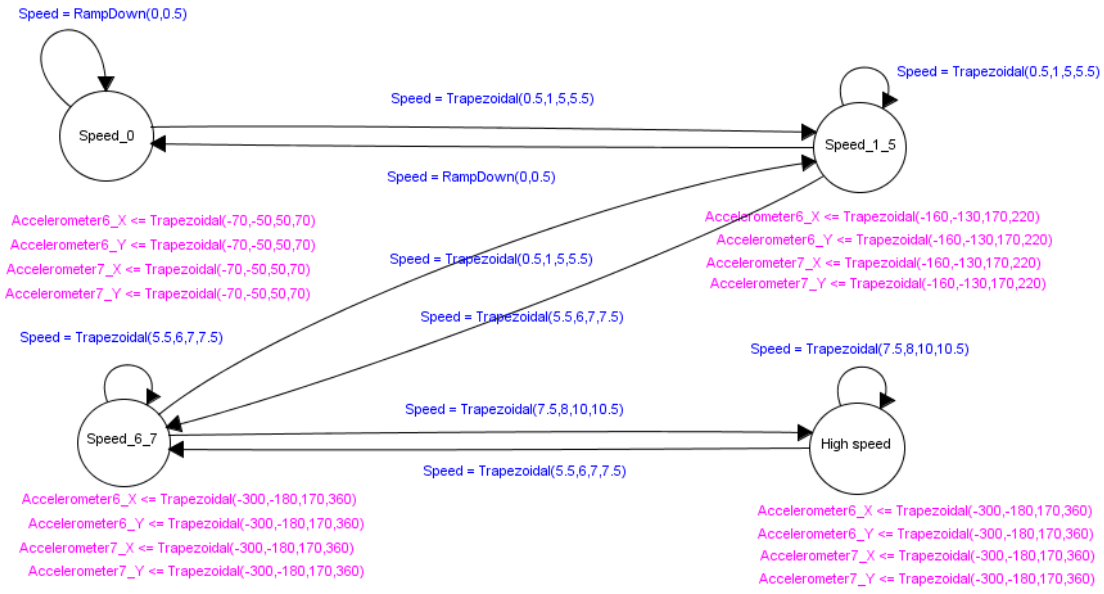

Figure 24: Behavioural Drift Analysis model for the mini-train

In the experiment, the train was started, increasing speed step-by-step, until the last wagon unhooked and derailed. As the speed continued to increase, the last two cars also unhooked and derailed. Before derailment, BDA measured a drift between the excepted behaviour and the observed one. The accelerometer when the wagons began to pitch too much in the bends. The sensors values of the nodes embedded on the cars where sending data every 250ms. Here is an example of a message received for nodes 6 and 7.

```
{
  "ServiceID": 100106,
  "Root": { "Gateway": 0,  "Source": 0, "TimeStamp": 0},
  "Nodes": [
  ...
    {
      "Safety": "true", "NodeID": 6, "TimeStamp": 1616060536, "TimeAccuracy": 759040355,
      "Sensors-Actuators": [
        {
            "SensorID": 3313, "TimeStamp": 1616060536, "TimeAccuracy": 759040355,
          "Resources": {"5702": -36, "5703": -1172, "5704": -1080, "5701": "m/s2"}
        },
        {
          "SensorID": 3336, "TimeStamp": 1616060536, "TimeAccuracy": 759040355,
          "Resources": {"5513": 0.0, "5514": 0.0, "5515": 0.0, "5518": 4246963}
        },
        {
          "SensorID": 3300, "TimeStamp": 1616060536, "TimeAccuracy": 759040355,
          "Resources": {"5700": -67, "5701": "dBi"}
        }
      ],
      "CRC": 2946992031
    },
    {
      "Safety": "true", "NodeID": 7, "TimeStamp": 1616060536, "TimeAccuracy": 915355443,
      "Sensors-Actuators": [
        {
          "SensorID": 3313, "TimeStamp": 1616060536, "TimeAccuracy": 915355443,
          "Resources": {"5702": -53, "5703": 1148, "5704": -1301,  "5701": "m/s2"}
        },
        {
          "SensorID": 3336, "TimeStamp": 1616060536, "TimeAccuracy": 915355443,
          "Resources": { "5513": 0.0, "5514": 0.0, "5515": 0.0, "5518": 4247236}
        },
        {
          "SensorID": 3300, "TimeStamp": 1616060536, TimeAccuracy": 915355443,
          "Resources": {"5700": -55, "5701": "dBi" }
        }
      ],
      "CRC": 2161284227
    }
  ],
  "CRC": 960448298
}
```


Figure 25: ITS message example

Figure 26 shows that the drift deviation started before the first wagon derailed. The too strong oscillation values of the last wagons lower the compliance of the observed values with the expected values. In straight lines, the wagons oscillate less and the drift value rises again before falling again during the following bends[2]. As soon as a wagon has derailed, the drift value remains at 0.

[2] https://youtu.be/9ABYxu37StA

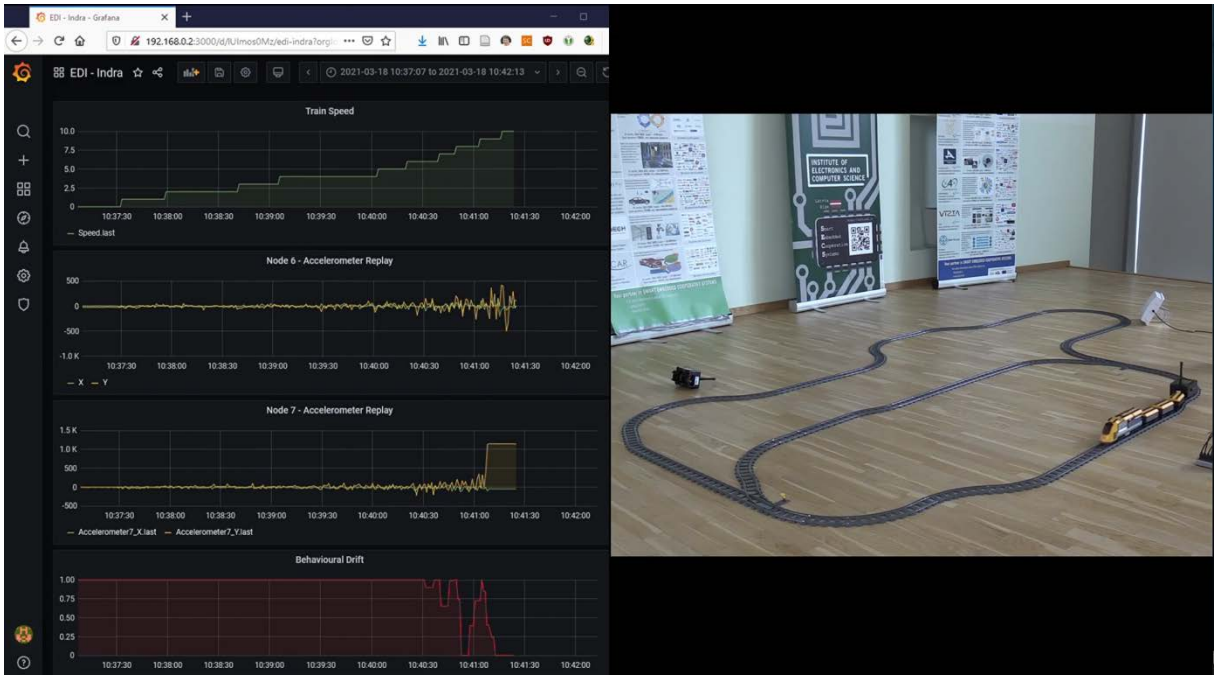

Figure 26: Behavioural drift analysis with the mini-train

A second use of BDA was carried out with the OpenBVE train simulator (Figure 27) in order to demonstrate its use in a wider context than the sensors and actuators available on the mini-train.

Two more complex behavioural drift models have been specified as presented in D1.4 with Figures 14 and 15. The first model specifies the safety conditions for the use of the train (opening of the doors is only allowed when the train is stationary). The second, more complex, specifies the train's compliance with the speed regulations that apply to it on the route. If there is a speed limit on a section of track, the model specifies that the train's behaviour drifts if this speed is exceeded (with a slight tolerance for exceeding the speed limit). This model thus makes it possible to demonstrate the implementation of tolerance zones when the specified data range is exceeded.

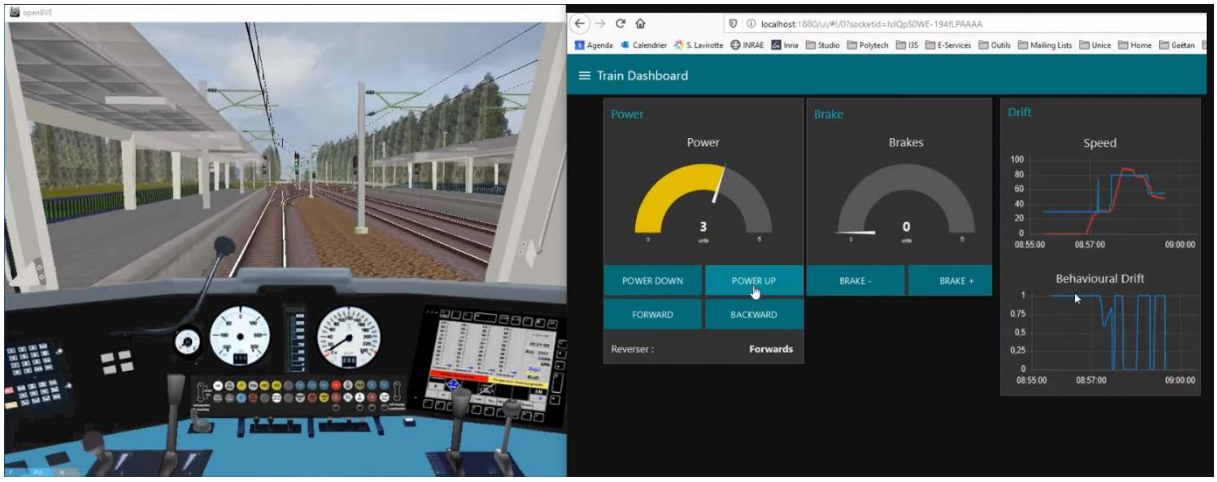

Figure 27: OpenBVE simulator and Behavioural Drift Analysis

Regarding the KPIs validation, TO2.1 is validated as the context monitoring and effectiveness analysis at operational time has been demonstrated in two different examples, one with the EDI mini-train and the messages coming from the ITS virtual infrastructure and one from the OpenBVE simulated train to extend and demonstrate the possibilities in a more complex scenario. We also note as one of the interest

of the tool is not only to provide a true or false value on the fact that the behaviour is respected or not, but a continuous value between 0 and 1 showing how close or far the behaviour is from what is expected.

2.1.1.6 Security and Privacy monitoring tool (S&P)

As it can be observed in Figure 28, Security and Privacy Monitoring and Control enabler (for short "S&P") with its associated S&P monitoring services was integrated and evaluated in the ITS use case. The integration required that the S&P backend was deployed in another Cloud infrastructure out of the ITS Virtual Infrastructure scope. The main reason for this is that the S&P enabler needs to be independent from the system being monitored, while still the S&P monitoring agents (in this case only one agent) require to be integrated physically with a piece of hardware that captures all the traffic of the real infrastructure.

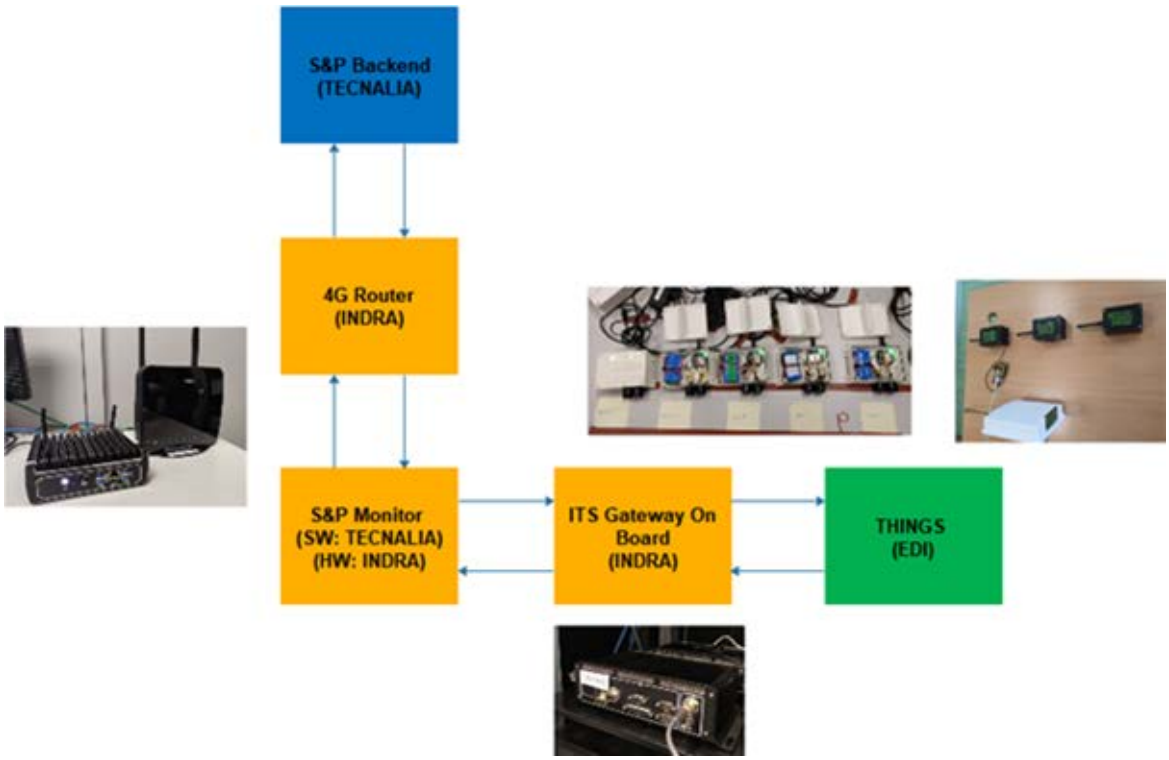

Figure 28: Architecture of ITS infrastructure with S&P enabler integrated

As it can be seen in the Figure 28, besides the S&P enabler backend a device dedicated to host the monitoring agent was included in the architecture designed and tested for the S&P tool. The Things require to be present also; therefore, the EDI Things are simulated using INDRA Off-the-Shelf devices as a messaging publisher. The physical ITS Gateway handles the traffic and routes it to the S&P Monitor agent which represents the S&P On Board part. This element is in charge of providing the On Board traffic data to the S&P Backend through a 4G connection to further treat the data and make the security analysis.

The S&P monitoring service is able to perform a security analysis and intrusion detection, so it can identify the users, the IPs and MAC addresses that are connected on board at any time. In case the tool detects unauthorised entities in the network communications, the S&P backend provides a notification to revoke the permissions of the user in the Figure 28 infrastructure, further details in section 2.1.3.4. The S&P enabler is also able to detect any abnormal behaviour of the traffic which will also be notified to the ITS system.

Next figures present the Dashboard of the S&P monitoring tool while monitoring and analysing the anomalies and attacks in the ITS on board traffic.

Figure 29 presents the overall monitor dashboard showing all the network protocols captured, being MQTT the one more relevant in this case. Figure 30 shows the alerts raised by the tool when a suspicious device with an unauthorised MAC address is detected. A similar alarm would be shown when the ID of the user of the gateway is an authorised user account. Finally, in Figure 31 it can be seen how the tool enables to detect anomalous traffic behaviour in the MQTT communications.

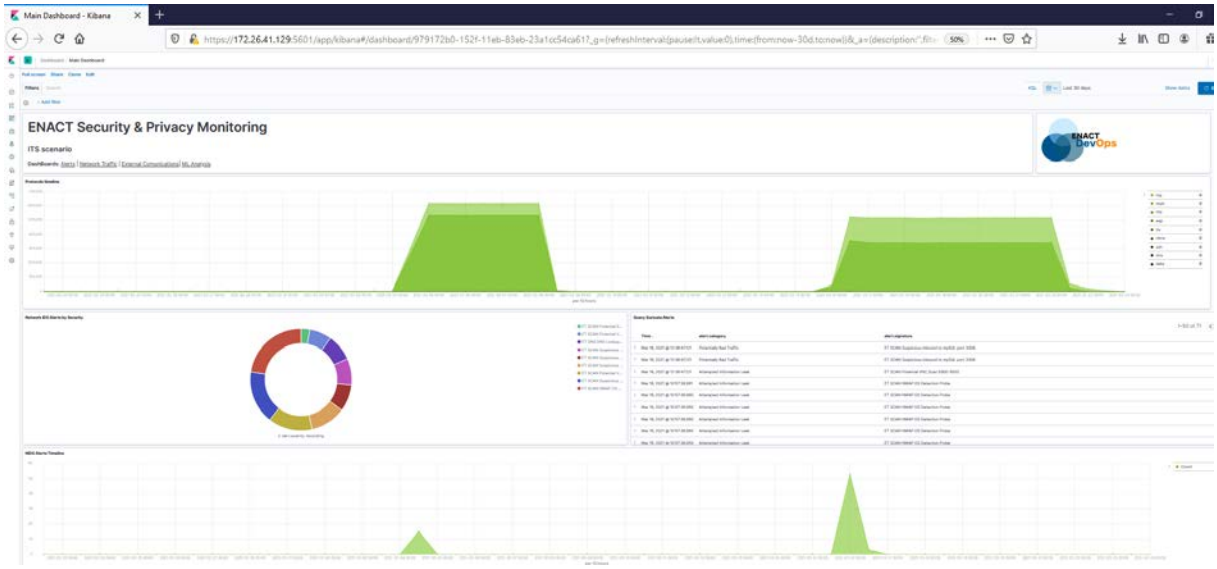

Figure 29: Network traffic dashboard of S&P enabler integrated in ITS use case

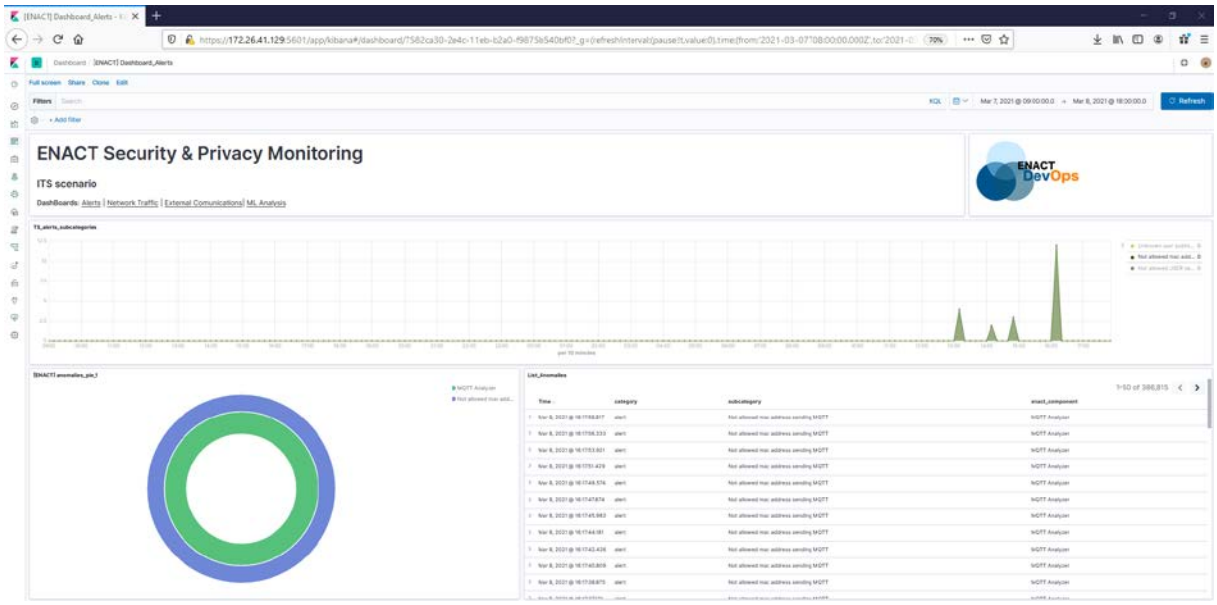

Figure 30: Detection of unauthorised device (MAC) by the S&P enabler integrated in ITS use case

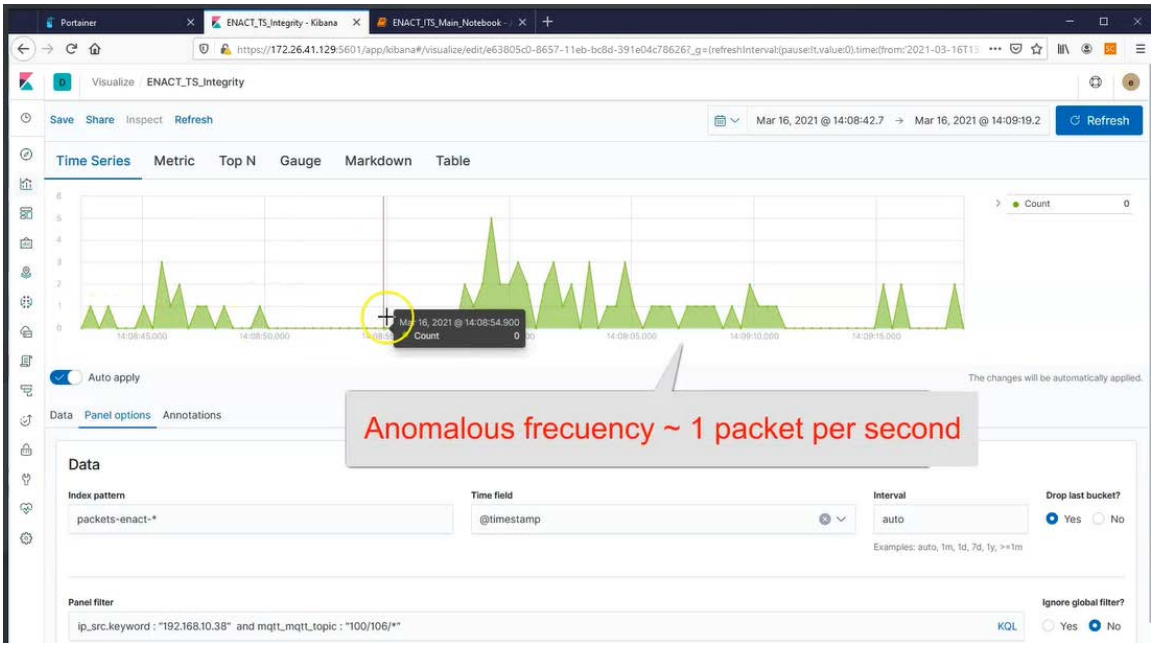

Figure 31: Detection of anomalous traffic behaviour by the S&P enabler integrated in ITS use case

As overall conclusion, the S&P tool has been validated as an effective tool that provides a complete security situational awareness of the on board communications in the ITS scenario (TO3.2). The main objective of the enabler is to make sure no intrusion can proceed, and that network traffic behaves as expected at all times, where accurate cadence of the different types of MQTT messages is being checked continuously. The key characteristic of the S&P tool is that it is agnostic to the system being monitored and thus is does not rely on any data coming from ITS system devices, but it produces its own analysis from data captured independently in the S&P monitoring agents distributed ad hoc (TO3.1). It is inferred from this then, that the S&P tool is also able to register the type of attacks that it detects (TO3.3).

2.1.2 Evaluation and Validation Application Scenarios

As it is stated in the deliverable D1.4, the ITS functionalities were tested and validated in the first period of the project. As a results summary, during the first period the main target was the integration of the EDI Things and the ITS Gateway. Previous this stage, the ITS Gateway developments, explained in the section 2.1.3, were internally adapted to the ENACT conditions while the EDI Things, based on the specification [5], are made with the ITS functionalities (Train Integrity and L&M).

The integration was made in several stages remotely using a virtual ITS Gateway. Once the Indra IRMS conditions are validated, as main security and safety standard reference for the ITS Use Case implementations, with the Things the ITS Use Case integration is validated in a remote manner.

After this remote integration, the physical integration was made changing only the ITS Gateway destination for the Things, from the virtual one to the real one, making minor configuration changes. It could be concluded, after validating again the IRMS conditions, that the ITS functionalities were full operative in a real scenario. The following Figure 32 shows the basic integration scheme and the Figure 33 with all the complete Clouds functionalities also defined along the deliverable D1.2.

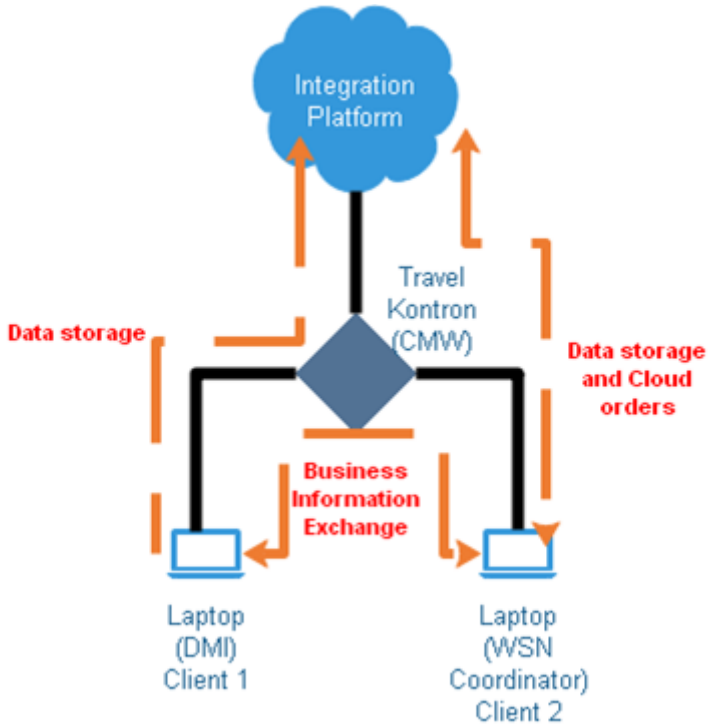

Figure 32: ITS Local Integration

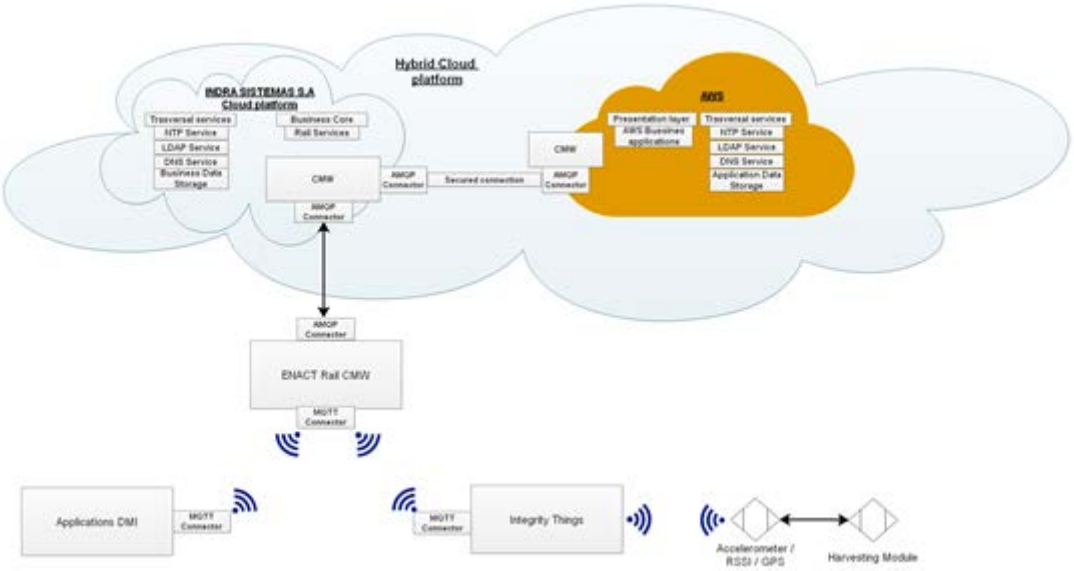

Figure 33: ITS Complete Interation Architecture

It can be concluded that the KPIs stated in the section 2.1.3 are completed, except the Metadata Report KPI. This KPI is validated with the integration results stated in the section 2.1.1.

2.1.3 Evaluation and Validation of the Business Case

The following Table 3 shows the state of the KPIs defined for the ITS Use Case itself.

Business KPI		Status
Main KPI		
INDRA, EDI, and BOSC ITS systems (Things, Gateway, and Cloud layers) integration and basic functionalities running.		DONE. The ITS subsystems has been integrated in a remote and in a physical manner with minor integration changes and the ITS data is available in a local and cloud environment to be served for the ENACT DevOps tools.
Sub-KPIs (required to achieve main KPI)		
Inauguration Process	Validation to set the wagons that form the composition	Completed. Validated by all the ITS participant partners (INDRA, EDI, and BOSC)
Integrity Process	Train Integrity process with a fixed composition	Completed. Validated by all the ITS participant partners (INDRA, EDI, and BOSC).
Logistic and Maintenance Process	Logistic and maintenance report attached to the composition presented	Completed. Validated by all the ITS participant partners (INDRA, EDI, and BOSC)
Metadata Report	Traffic metadata to evaluate the system status	Completed. Validated by all the ITS participant partners (INDRA, MONTIMAGE, and SINTEF)

Table 3: ITS - Business KPIs

As a summary, it can be stated that the 100% of the works related with the ITS Use Case itself has been accomplished.

2.1.3.1 ENACT DevOps tools Integration

This section is intended to provide the context of the developments carried in the second period of the project. At the end of the first period, the scenario defined in the deliverable D1.1 (Scenario 1: Logistics and Maintenance) were 100% functional and operation. The main developments, as stated in the section 3.1.1, for the second period are related with the metadata reports about the hardware, connectivity indicators, and operation status of the ITS infrastructure. This functionality was mainly validated internally as the following Figure 34 shows. The Figure 34 shows that the gateway are exchanging the messages in a running Federation (RabbitMQ mechanism to connect RabbitMQ servers) in the right side. In the left side another RabbitMQ server receiving the messages with a specific rate depending on the servers connected.

Figure 34: ITS Gateway Status Validation. Source: INDRA

Several internal tests have been performed to evaluate the functionality previous been integrated with the ENACT DevOps tools. The tests are based on:

Changing the state of the system functionality to check that the operation state (device heartbeat) includes the current operational state.

The ITS Gateway load is modified with different messaging charge, as it is done in the T&S tests, to check that the hardware and connectivity parameters variate in a coherent manner with the testing load injected into the gateways.

This functionality has been used with the GeneSIS, RCA and T&S tool. Further implications into the test are explained into the ENACT tools evaluation explained into the sections (3.1, 3.2, and 3.4).

2.1.3.2 Routing Topologies

The ITS Gateway is mainly specified both as a messaging broker (delivery and transport of messages towards their predetermined destination) and a message processor (transform and adapt the message contents to its recipient). These two general functionalities are what are considered the "business-focused" main capabilities of the current implementation.

Each general functionality has its own configuration, but all configurations are dependent of one another and must cooperate in order to deliver a working system. It is required that these both functionalities are defined in a separate manner to the routing topology.

The routing topology is the way in which elements are connected through links in order to make all destinations available and reachable from different origins. It has been observed that the DevOps dynamic introduces several entities that are dependent or independent of each other. This dependencies definition must be defined separated to the gateway core and server functionalities to not affect them and tackling the deployment in a more reliable and agile manner.

Each layer or level of the gateway has a specific topological design that depends on the rail service criteria followed. In this case, there are different topological designs in a same level depending on the service, or set of messages, offered. The whole deployment for a service in a layer is called virtual host. A virtual host is defined per services and includes all the rules that must be followed to connect the entities that participate on that services. This makes the services definition independent of the core functionalities and service independent. Moreover, it makes the services easy to monitor and to maintain.

2.1.3.3 Gateway Status and Operational Stage Provision

The ITS Gateways are only able to authenticate and to route messages from the clients. However, the ITS Gateway does not act as a client itself. It is identified for several ENACT tools that the status and the operation state of the ITS Gateway must be known in order to evaluate the ITS infrastructure and, then, introduce this ITS infrastructure into the DevOps philosophy. It is not possible evaluating only the information routed by the ITS Gateway, it is required that the gateway publishes its own information

and it is not possible with the development status of the ITS Gateway made in the first period of the project.

During the second period, a set of refactoring tasks have been tackled in order to make the first period functionalities and the functionality that enables the ITS Gateway to publish its own events working together. This refactoring also considers the fact that the development results must be included into Docker images to be deployed.

Once the refactoring is evaluated, the software functionality that permits to the ITS Gateway publishing its own status and operational stage is possible. This piece of software is able to track the service operational status (e.g. the train integrity system is running, the logistic and maintenance service has stopped,…) and also publish its own status information tracking, in real time, the hardware and connectivity metrics of the ITS Gateway. The following metrics are considered:

Connectivity metrics:

a. Number of gateways up connections: It defines the number of connection interfaces up in the monitor moment.
b. Published message size: Average size in Bytes of the published JSON files received in the interface.
c. Received message size: Average size in Bytes of the received JSON files received in the interface.
d. Number of published messages per second: It defines the number of JSON files published into the connection interface.
e. Number of received messages per second: It defines the number of JSON files received into the connection interface.
f. Message Delay: Average delay in the message's reception. It is defined as the difference between the gateways timestamp marked in the transmission gateway and the timestamp marked by the received gateway (resolution in milliseconds).

Hardware metrics:

a. CPU usage: CPU average usage among all the cores.
b. RAM usage: RAM average usage among all the device RAM cards.
c. Power consumption: Device power consumption (DC) in Watts.
d. Disk usage: Disk average usage among all the device disks.

Operational metrics:

a. Services States Machine: Indicates the status of the gateway related with the functionality that it manages. (S00 in the Train Integrity functionality).
b. Timer: Indicator of the periodicity to publish these heartbeat message.
c. Operational Status: Provides the operational status of the ITS Gateway to enable security capabilities to get connected by partners.

As it was mentioned in the previous sections, the status of the ITS Gateway is used by the RCA tool to evaluate how are the ITS Gateways performing and, in case of it is detecting predefined tool trained failures in the ITS infrastructure, locate that ITS Gateways that are causing this failure. Moreover, the Testing and Simulation tool is able to evaluate the performance limit of an ITS Gateway by tracking its status and inferring its limits to know how to scale it into the infrastructure. A final usage that is extracted from this functionality is inferring the operational state of a system and get information about when it is possible making a Docker image deployment. As soon as a gateway is running a functionality, it is not possible performing developments.

As it was mentioned in the previous sections, the status of the ITS Gateway is used by the RCA tool to evaluate how are the ITS Gateways performing and, in case of it is detecting predefined tool trained failures in the ITS infrastructure, locate that ITS Gateways that are causing this failure. Moreover, the Testing and Simulation tool is able to evaluate the performance limit of an ITS Gateway by tracking its

status and inferring its limits to know how to scale it into the infrastructure. A final usage that is extracted from this functionality is inferring the operational state of a system and get information about when it is possible making a Docker image deployment. As soon as a gateway is running a functionality, it is not possible performing developments.

2.1.3.4 ITS User Revoke Process

The User Revoke Process is a mechanism that disables a user account in ITS system whenever an alert is raised by the S&P enabler. The revocation is the reaction mechanism to rule-based detections by the S&P monitoring service about users not following the expected behaviour. The alerts are transmitted from the S&P enabler to the Kafka bus in the S&P backend and the ITS system is subscribed to them. The subscription mechanism software receives the alert notification in Kafka, which includes the identification of the user to be revoked.

Once this notification is received, the mechanism blocks the user account in the ITS Central Authentication Server; therefore, the user cannot publish into the infrastructure anymore.

2.1.3.5 FIWARE

For the ENACT project, an integration between the FIWARE containers and the ITS infrastructure is performed. This section is intended to summarize the usage of the FIWARE components into the INDRA ITS infrastructure. The following Figure 35 shows a general implementation of the INDRA and FIWARE systems.

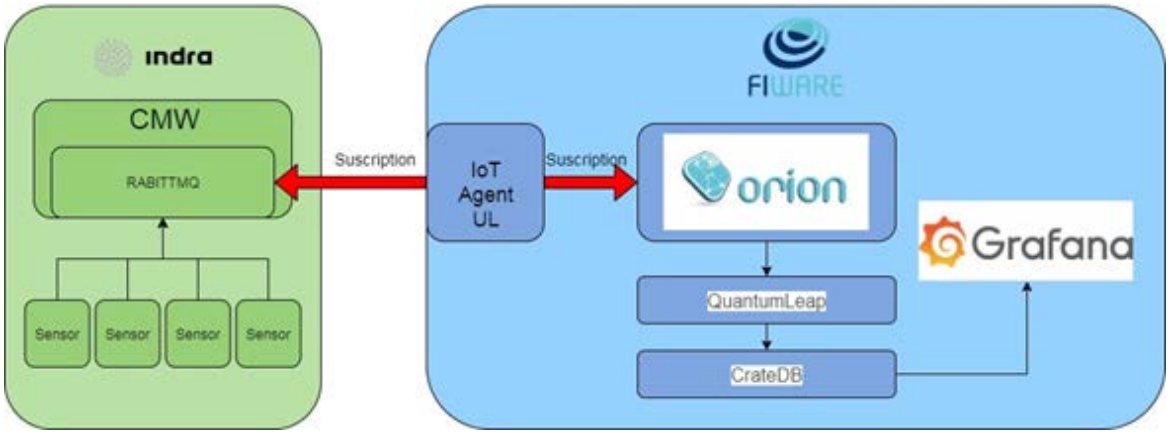

Figure 35: ITS - FIWARE Concept Scheme

and the following Figure 36 shows a more detailed need for the components required:

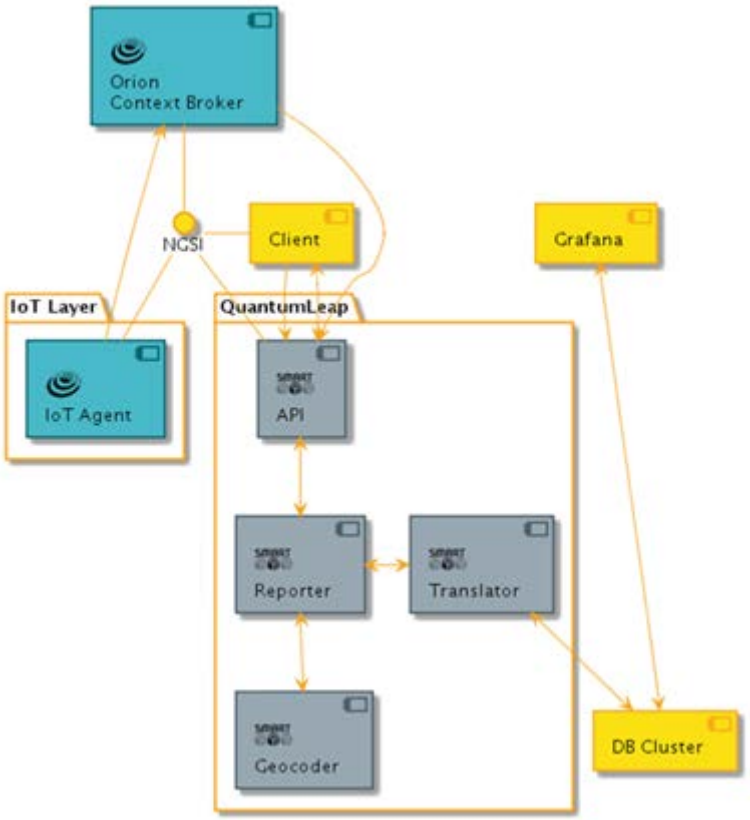

Figure 36: ITS - FIWARE Components

The next sections show the different required details for every component referred at the Figure 36. In any case, the next link shows an example that integrates all the steps required to report data from the Orion broker to the Grafana representation:

IoT Agent UL Component:
This Internet of Things Agent is a bridge that can be used to communicate devices using the Ultralight 2.0 protocol and NGSI Context Brokers (like Orion). The Internet of Things Agent that uses Ultralight 2.0 protocol as AMQP, HTTP, and MQTT transport protocols covered; therefore, the IRMS solution is compatible with FIWARE's brokers as a valid IoT data source agent. This IoT Agent is designed to be a bridge between Ultralight and the NGSI interface of a context broker. It is based on the IoT Agent Node.js Library.

QuantumLeap Component
The QuantumLeap component acts as a database at FIWARE in the ITS infrastructure. The first need is having QuantumLeap and its complementary services running. After the installation, it is required to connect the QuantumLeap component with the Orion Context Broker through an NGSIv2 subscription for each entity type that requires to be stored. Historical data for each entity type will be added in the QuantumLeap's database as long as the subscription is active, correctly configured and the entity in the notification is NGSI compliant. QuantumLeap supports both Crate DB (explained in next section) and Timescale as time-series DB backends.

At the moment, the only actively supported distribution of QuantumLeap is based on Docker. It can be built and installed from sources. Reusing Orion services, testing at local or deploying it in a Docker manager is possible.

CrateDB Component
CrateDB has been chosen as the time-series data sink for QuantumLeap, because, among many other benefits, it integrates seamlessly with the Grafana time series analytics tool. Grafana can be used to display the aggregated sensor data.

2.1.4 Conclusions

At the beginning of the project, 7 tools were defined to be integrated. After the D1.1 analysis, the Online Learning Techniques, were removed from the ITS Use Case as its scope do not accomplished the ITS objectives for it, in this project, as it is justified in the after D1.1. The other 6 tools were integrated and the ITS Use Case adapted successfully to them. The main benefits that these tools add to the ITS Use Case are defined in detail into the D1.4. From this D1.4 impact analysis we can extract the main conclusions:

"IoT services in the domain of train integrity control, in particular for the maintenance and logistics of the rolling stock and the on-track equipment." All the functionalities are 100% operational.
"*End device connected to sensors: initially up to 3 different WSNs will be installed along the train. Each WSN has a WSN propagator node. This end device will be directly connected to sensors or to any other systems that helps to verify train integrity."* The sensors are provided and used in the mentioned Train Integrity and L&M functionalities.

"Enabling predictive maintenance and increasing safety of Rail IoT services using the context data that cannot be obtained by wired sensors, based on the ENACT trustworthiness toolkit.

"Gateway: one Gateway to manage all Wireless Sensor Nodes (WSN)". A single gateway can manage the 100 devices as it is shown in the section 4.2. Proving the scalability factor of the state ITS infrastructure.

"EDI and BOSC will provide 100+ node wireless sensor modules (WSM) TestBed." This test bed could be simulated and handled by the gateway.

Regarding the specific validation and verification report objectives for the ITS Use Case, the main conclusions are listed below:

- The GeneSIS tool is able to deploy software versions into the gateway layer to enhance its security capabilities or/and deploying a new version of the software without interfering with the previous software deployed.
- The RCA tool is able to detect and report pre-trained failures that affect to the communications into the gateway infrastructure and with the edge layer.
- The T&S tool is able to provide the scalability capacity a gateway.
- The ACM tool is able to prioritize orders in a rail environment.
- The BDA tool is able to model the Things behaviour, enhanced with an external simulator of the real Things, and detect deviations.
- The S&P tool is able to monitor the security and safety capabilities of the On Board infrastructure.

2.2 Digital Health

This section presents the validation and verification of the ENACT enablers in the Digital Health use case. During the first period of the project, several ENACT tools were successfully tested. In particular to enable continuous deployment separating between the initial Software deployment and Gateway on boarding and the continuous deployments as part of the system evolution, context aware access control of edge and IoT devices, and risk management as an integral part of the DevOps process to ensure compliance requirements in an efficient way. During the second period, these has been further explored and evaluated and additional evaluations has been carried out related to test and simulation

The Evaluation and Validation of ENACT results in the first period of the project applying the Smart Building use case has included:

- Evaluation and validation of the use case itself as a reference for the ENACT tools and a set of the first versions of those tools in a series of DevOps scenarios. The tested DevOps scenarios focused on the orchestration and continuous deployment across the IoT, edge and cloud space trustworthiness, and risk management.
- Evaluation and validation of the Orchestration and Continuous Deployment, in particular the ThingML tool as well as some exploration of the DivEnact and GeneSIS tools.
- Evaluation and validation of the Context Aware Access Control tool which in the first phase was applied and evaluated related to manage context aware access control of edge and IoT devices interactions.
- The risk driven decision support was explored in the first phase to ensure proper risk management as an integral part of the DevOps process.

In the second period these enablers and tools was further evaluated and validated with more scenarios and further tools were evaluated. In particular, in the second period the digital health use case has been applied to perform evaluation and validation of:

- The Orchestration and Continuous Deployment enabler bundle with the ThingML, GeneSIS and DivEnact tools,
- the Context aware access control tool applied for more scenarios including business level scenarios addressing context aware access control of different stakeholders according to context (e.g., further access can be granted in case of an emergency situation, for example, providing access to an eHealth supervision camera in a patient home for the fire fighting department in case of a fire),
- The risk driven decision support tool where scenarios related to management of risks concerning privacy has been included
- Test and simulation tool has been evaluated related to more efficient testing of the Digital Health use case.

A summary of the evaluation and validation performed applying the Digital Health use case is shown in Table 1, including both first and second period results. The summary is provided in the context of the identified KPIs related to the Digital Health use case (as stated in D1.1).

Code	KPI	Enabler involved	Unit of measurement	Who	Status

TO1.1	**Continuous deployment across the IoT, edge and Cloud space.** (i) Include at least 2 IoT&Edge nodes and 2 cloud nodes. (ii) 10 deployments (iii) Includes orchestration, setting up interoperation with "no downtime"	Orchestration and Continuous Deployment enabler	# of IoT, edge and cloud nodes # of deployments "Zero" downtime	Tellu, SINTEF	Evaluation based on ENACT enablers and integrated with ENACT supported state of the art technologies (including Ansible, Jenkins, GitHub, Docker, Kubernetes). In particular ThingML is applied for all code development of the PHG, both for the code handling application functionality and code for managing DevOps (e.g., GW agent, monitoring probes etc), and the ENACT Deployment bundle (ThingML, DivEnact and GeneSIS) is applied for continuous deployment. **More than 10 simultaneous deployments** are performed on **IoT&edge nodes** (the personal health gateway (PHG) and Arduinos) exceeding the KPIs of at least 2 IoT&edge nodes, 2 cloud nodes and 10 deployments. It includes orchestration and setting up interoperation with **no "down time"**. Deployments includes update and deployment of software on PHG and IoT devices connected to it.
TO1.2	**Automatic change or upgrade.** Demonstrate 5 changes or upgrades of 5 independent Gateways. Performed automatically and without need of physical intervention (or minimize physical intervention)	Orchestration and Continuous Deployment enabler	# of changes, # of Gateways # of physical interventions	Tellu, SINTEF	10 **independent Gateways** with **several changes**, performed automatically and **without need of physical intervention**. Applying ThingML and baseline technology stack of the Orchestration and Continuous Deployment enabler.
TO1.3	**Automatic Pairing of devices with Gateway after reset.** At least 5 different devices automatically paired with Gateway after reset	Orchestration and Continuous Deployment enabler	# of different devices	Tellu, SINTEF	Evaluation performed with **5 different devices** automatically paired with Gateway after reset. Applying ThingML and baseline technology stack of GeneSIS (the Orchestration and Continuous Deployment enabler)
TO1.4	**Simulation of at least 5 different devices/things**	Test, Emulation and Simulation enabler	# of supported devices	Tellu, MI	Evaluation and validation performed simulating the Personal Health GW and sensor inputs and the cloud backend. More than 5 devices simulated, in general arbitrary number of devices can be simulated
TO1.5	**Combination of simulated and physical devices** Demonstrate a combined test environment that apply at least 5 different simulated devices and at least 3 different physical devices	Test, Emulation and Simulation enabler	# of devices	TellU, MI	Combination of actual and simulated PHGs and corresponding sensors has been evaluated
TO3.1	**Recovery after failures** Demonstrate the successful recovery after injected failures of two gateways	Robustness & Resilience enabler	Y/N #Time to recovery	TellU, SINTEF	Successfully tested in the emergency access control scenario (Section 3.2.2): For those gateways that are deployed with new version of software with alarm-controlling logic, when the alarm button is unplugged from the gateway, the default version without alarm control logic will be deployed to the gateway. The deployment is fully automatic, controlled remotely by the DivEnact service, without affecting other gateways. Time for recovery: 3-5 minutes.
TO3.2	**Configuration Rollback** Demonstrate configuration Rollback	Robustness & Resilience enabler	Y/N	TellU, SINTEF	Successfully tested in the same dynamic access control scenario, in the following two perspectives:

					1. Device-level rollback: After deploying the new version of software with alarm-related logic to a gateway, if developer choose to roll back the configuration on this gateway, the previous deployment will be deployed automatically to the same device, replacing the new configuration. 2. Fleet-level rollback: if developers choose to rollback the software version with the alarm-controlling logic, all gateways with this version of software deployed will rolled back to the previous version of software.
TO3.3	**Diverse deployments** Demonstrate the automatic deployment and maintenance of at least 5 different deployments among the customers	Robustness & Resilience enabler	# different deployments	TellU, SINTEF	Successfully tested in the emergent access control scenario with 5 different versions of software: the basic version without alarm control, and four different version of alarming controlling software using Arduino-based button, button on SensHat, simulated button, and a virtual button triggered by temperature. All versions are automatically distributed to the gateways according to their context and capacities.
TO3.4	**Automate risk management** Demonstrate automatic risk management by integrating ENACT risk management functionalities into mainstream DevOps software for seamless tracking and monitoring of risks related to security, privacy and trustworthiness	Risk Management enabler	Demonstrate Automation	Tellu, BEAWRE	Successfully testing of the automatic risk management integrating the tool with widely applied DevOps issue management tool Jira. The tool support modelling of the system components and the information flow and provide automatic generation of vulnerabilities risk and mitigations based on a build in expert system.
TO3.6	**Protection of person sensitive data** Demonstrate protection against privacy attacks and incidents	CAAC, S&P Control enabler	Y/N	TellU, EVIDIAN, Tecnalia	These mechanisms is provided in the CAAC enabler. CAAC provide the ability to adjust risk level to dynamically adjust the protection level in order to avoid incidents, mitigate attacks etc
TO3.7	**Access control** Demonstrate the control of access to the gateway to authorised users only Demonstrate the control of access to the gateway to authorised services only	CAAC, S&P Control enabler	Y/N Y/N	TellU, EVIDIAN, Tecnalia	Validated applying the Context Aware Access Control (CAAC), in particular the device authentication ensures access to authorised users and services. The latter was validated in the first phase of the project and demonstrated in the first review. The context aware access control of the users has been validated in the last phase of the project

TO 3.5 & TO3.8-TO3.10	**TO 3.5: Real-time monitoring of a set of Medical Gateways and reception of proper notifications with useful information in case of errors.** **TO 3.8 Secure data transmission** Demonstrate the enforcement of use of Secure protocols **TO 3.9 Trustworthy communications in the sense of reliability, availability, integrity and privacy** Demonstrate the availability of enforcement controls **TO 3.10 Monitoring, Diagnose information and failure detection** Detection and Diagnosis of suspicious behaviours at cloud level	S&P Monitoring enabler	Y/N	TellU, Tecnalia	Initial experiments conducted based on ENACT enabler principles and supported by state of the art tools (in particular Grafana and Prometheus). Based on these experiments and the evolution of the S&P Monitoring enabler, it appeared that the proposed enabler is not compatible with the security architecture of the Digital Health use case. For the Digital Health use case it is significant to have strong protections of the data as it to a large extent is person sensitive (health data etc). Thus, all the communication is required to be encrypted and remote access from external services or tools for sniffing/monitoring data are not allowed to adhere to security requirements. As the S&P Monitoring enabler required such access and also required to receive unencrypted application data/metadata, the Digital Health use case is not suitable for validating the S&P Monitoring enabler. Instead, this enabler has been validated applying the two other use cases (according to the principles of performing validation on at least two use cases)

Table 4: Digital Health - ENACT Tools KPIs

2.2.1 Evaluation and Validation DevOps Scenarios

Group of tests	Test	Description	ENACT tool	Test result
Initial Software deployment and GW onboarding	Test 2.1.0.1 Development/Specification of Generic image Test 2.1.0.2 Initial deployment of generic image Test 2.1.0.3 GW onboarding	Test of ENACT development/specification toolchain, initial deployment and GW onboarding,	Orchestration and Continuous Deployment enabler	Test completed, result reported in TO1.1 in the previous section, and in the related evaluation and validation scenarios described in the next section. ENACT Deployment bundle (ThingML, DivEnact and GeneSIS) integrated with state of the practice technology stack of the enabler (Ansible, GitHub, Jenkins, Kubernetes, Docker).
Software deployment and evolution in the Gateway	Test 2.2.0.1 Software deployment and evolution in the Gateway	Test of software deployment and software evolution across edge and cloud (GW and cloud server).	Orchestration and Continuous Deployment enabler	Test completed, result reported in TO1.1 in the previous section and in the related evaluation and in validation scenarios described in the next section. ENACT Deployment bundle (ThingML, DivEnact and GeneSIS) integrated with state of the practice technology stack of the enabler (Ansible, GitHub, Jenkins, Kubernetes, Docker).
Gateway recovery and factory reset in the field	Test 2.3.0.1 Gateway recovery and factory reset in the field	Tests the capability of remote recovery and reset functionality of potentially huge number of edge components accounting for both their physical and virtual nature.	Orchestration and Continuous Deployment enabler & Diversifier enabler (DivEnact)	Test completed. Results are described in TO3.1， TO3.2 and TO3.3 in the previous section.
Software testing on the Gateway	Test 2.4.0.1 Software testing on the Gateway	Test of ENACT capability of performing efficient testing across IoT, edge and cloud space	Test, Emulation and Simulation enabler & Risk Management enabler	Test completed. Results are depicted in TO 1.4 and TO1.5 and in validation scenarios described in the next section.
Continuous integration of gateway modules and versioning	Test 2.5.0.1 Continuous integration of gateway modules and versioning	Test of DevOps continuous integration capabilities and versioning across IoT, edge and cloud	Test, Emulation and Simulation enabler, Orchestration and Continuous Deployment enabler	Test completed. Results are depicted in TO 1.1, TO 1.4 and TO1.5 and in validation scenarios described in the next section.
Trustworthiness of the medical system	Test 2.6.0.1 Trustworthiness of the medical system	Test of ENACT capabilities of efficient support for ensuring proper security and privacy of the full DevOps process according to customer requirements as well as regulations and laws	Security and Privacy Monitoring and Control enabler, Risk Management enabler Robustness & Resilience enabler	Test completed. Results are depicted in TO 1.4 and TO1.5 and in validation scenarios described in the next section.

Table 5: Digital Health - ENACT Tools Validation and Verification Test Progress

2.2.2 Evaluation and Validation Application Scenarios

2.2.2.1 The ENACT deployment bundle and context aware access control scenario

We tested the ENACT enablers (ThingML?) GeneSIS, DivEnact and CAAC using the real Tellu production set up, which illustrate how the fleet deployment bundle works in the DevOps process and what it means to different stakeholders of the RPM product.

For the current RPM product, Tellu has distributed 200 gateways, each of which is used by a single patient. Tellu operates all the gateways, on behalf of different customers, i.e., nursing service providers running by either private companies or local governments. All the gateways collectively form a large and geographically distributed fleet for Tellu.

The testing scenario is introducing a new access control mechanism into the RPM fleet. The RPM service monitors the real-time status of a patient, and it requires strong access control to protect the data privacy for the patients. Among the sensors, the cameras are used for the most sensitive data, i.e., the live video of the patients and their living environment. In the current system, the access control strategy is strict and static: only the personal nurse can access the live video of a patient, during specific time slots in the day. Recently, one tenant raised a new requirement for a more flexible access control strategy: Under emergency conditions, such as when the patient falls, the nurse on duty should be granted with a temporary access to the camera to check the situation around the patient.

To achieve this, Tellu first needs to integrate Evidian's new context-aware access control service, i.e., the CAAC enabler. The enabler is a SaaS maintained by Evidian. For each request to the device data, the Tellu backend service invokes the CAAC service to decide whether the request should be authorized. CAAC extends the tradition access control service with a separate service to maintain the context of the data providing device, such as the risk level of the device, and consider this context when making the authorization decision.

The DevOps team in Tellu decided to implement this feature gradually, starting from the basic emergency situation when the patient presses an alarm button. The button hardware includes a micro-controller board (Arduino Uno) connected to the gateway via a USB port, and a remote Bluetooth-connected button that the patient has to wear as a necklace. The software part includes: (1) C code running on the Arduino board to listen to the remote button, (2) an updated version of gateway software to bind the button with the patient and to upload the button click event to CAAC's context service.

Tellu has started to distribute the button hardware to voluntary patients, and need to deploy the updated software to the gateways that belong to the requesting tenant and has the button hardware attached. This is a typical fleet deployment problem: the software variant with new access control exists together with the previous version, and should only be assigned to a sub-set of the gateways depending on the tenant and hardware contexts. The "last-mile" deployment of the assigned software to the gateway involves software deployment on heterogeneous devices, including micro-controller boards without an operating system and an independent network connection.

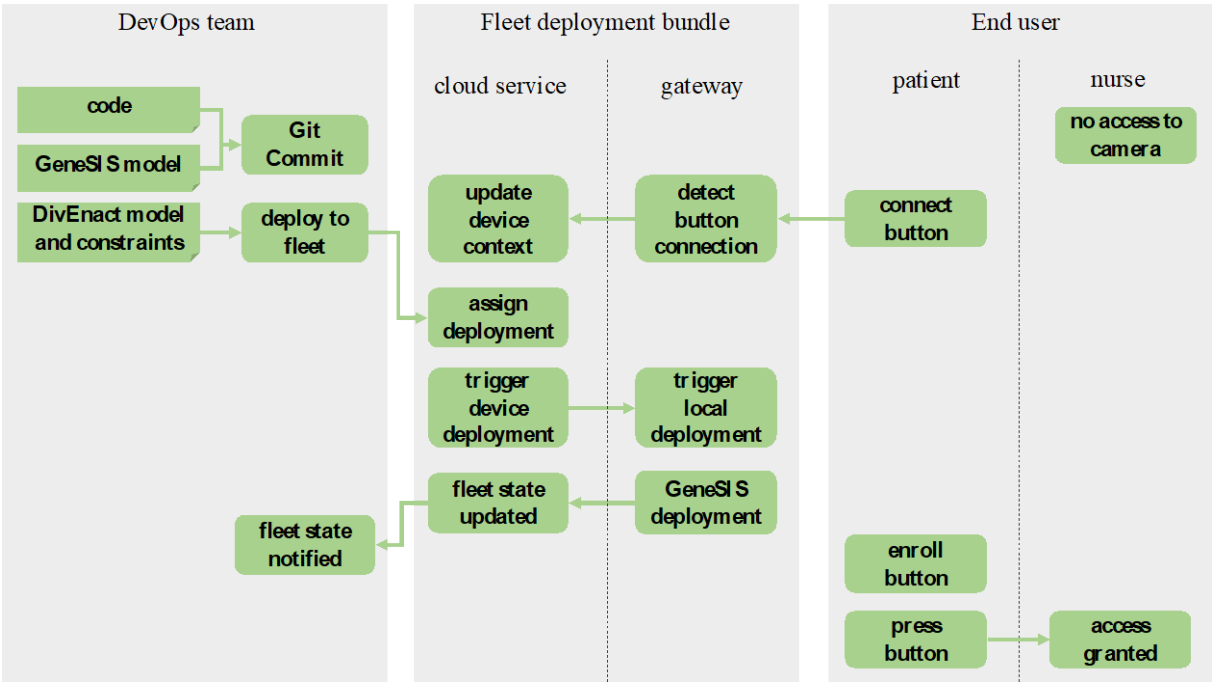

Figure 37: Using DivEnact and GeneSIS to dynamically introduce the CAAC function into Tellu gateways

Figure 37 shows how the fleet deployment bundle is used to realize this scenario. In the diagram, the corner-folded boxes represent the input artefacts, and the rounded rectangles are the activities. The arrows represent the causal links between activities, i.e., the target activities happen as a result of the source one. The vertical spatial relationship between activities roughly indicates their temporal relation, i.e., the activities placed higher in the diagram happens before the lower ones. We now describe the usage from three different, yet closely related perspectives - namely, the DevOps team, the components inside the bundle, and the end users.

From the perspective of the DevOps team. The Tellu DevOps team uses the DivEnact GUI to manage the fleet of devices. Figure 38 shows the device management tab of the DivEnact GUI, with sample fleet comprising physical and simulated gateways. The map view on the right shows the locations of the physical gateways placed in SINTEF (Oslo), Tellu (Asker, Norway), CNRS (Nice, France) and ISRAA (Treviso, Italy). A mark in the map may represent multiple devices. The locations are derived from the IP address of the gateways, and therefore are not precise – it shows the location of the ISP that provide internet access to the gateways. Such course-grained location is by design, so that the DevOps teams can have an intuitive view of which customer the devices belong to, without knows the precise locations of the patients.

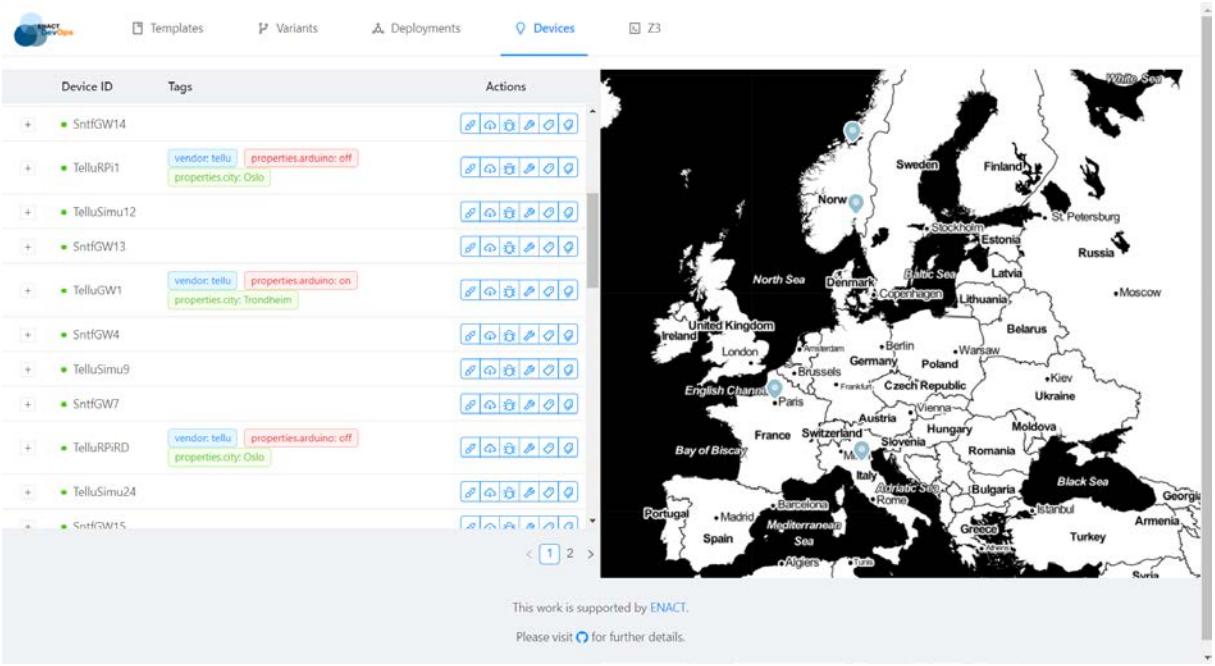

Figure 38. Sample fleet for the demo scenario

The development phase for this scenario yields three artefacts: (1) Source code running on the gateways and micro-controllers, which are further built into a Docker image and a code package; (2) GeneSIS deployment model describing how to deploy the Docker container and the code package into abstract gateways and micro-controller; (3) fleet deployment models and constraints. In this scenario, the important constraints are:

- The new deployment should only apply to the gateways that belong to the specific tenant;
- It should not be deployed to gateways without the button hardware.

The first two artefacts will be committed to the Git repository as part of the software, while the third one is the input for the DivEnact tool to configure the fleet assignment activity. Figure 39 shows a sample shows the DivEnact model for this scenario: Starting from a template with all the required containers in the first screen shot, followed by a variant defined from this template, with the current GeneSIS for the Arduino code.

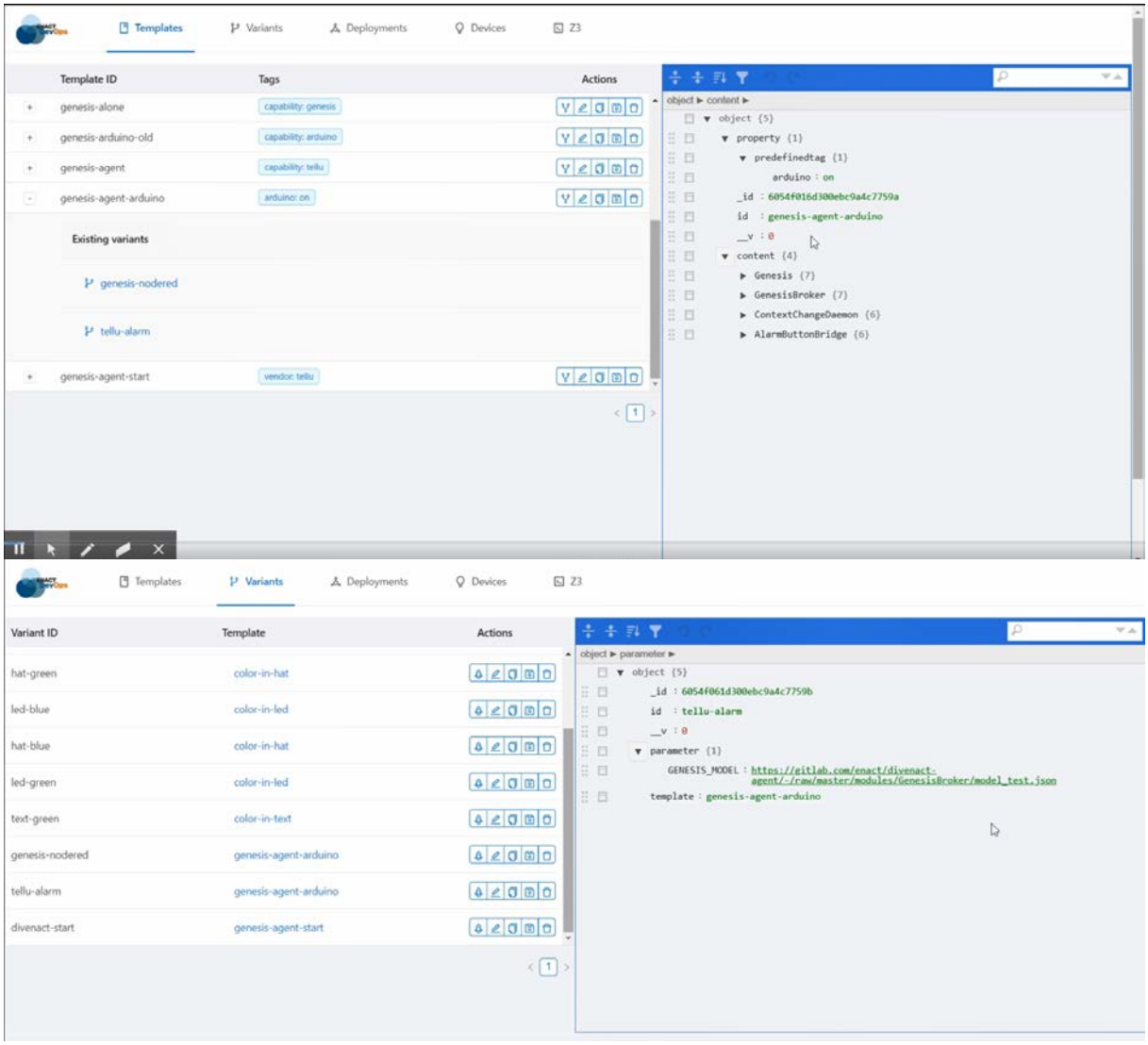

Figure 39. DivEnact fleet deployment model

Once the artefacts are ready, the DevOps team will launch fleet deployment action, by clicking the "generic" deployment button from the variant (Figure 40). It is worth noting that during the whole process, the DevOps team does not need to know which concrete devices belongs to the tenant, and whether and when the patients have plugged the button hardware. They only view the entire fleet as a whole.

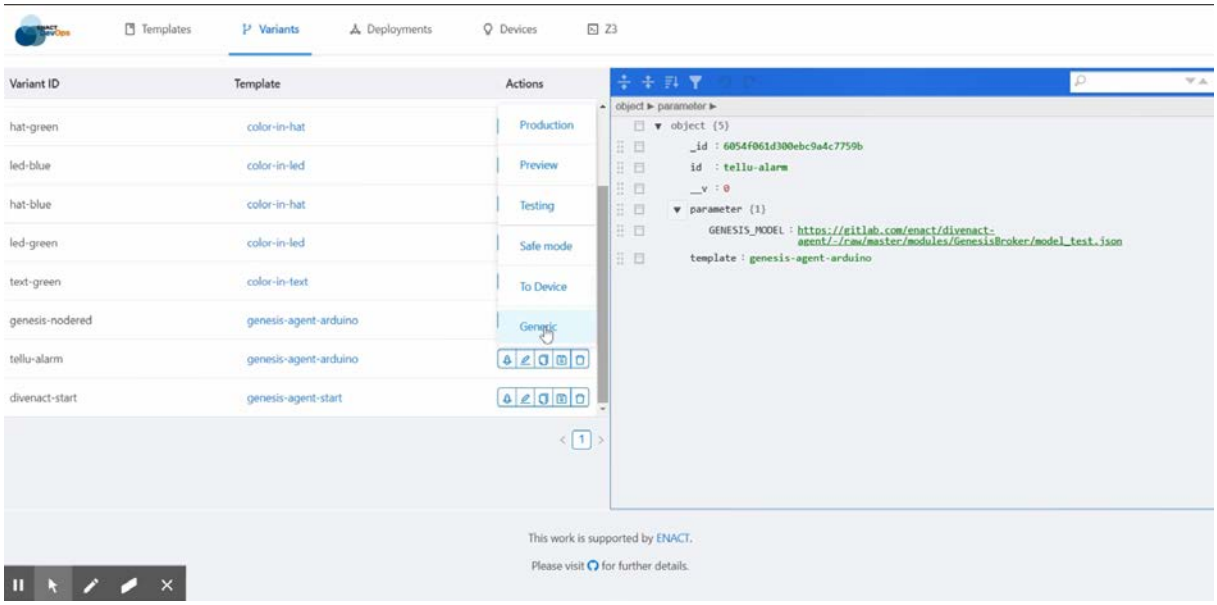

Figure 40. One-click deployment of the DivEnact model

Inside the fleet deployment tools. The components within the fleet deployment tools are running in two different places: the cloud and the edge gateways. The agents in the gateways keep detecting new hardware, and update the context of the gateway managed by the cloud service. Once fleet deployment is triggered by the DevOps team, it will first perform fleet assignment to decide which gateways should be provisioned with the new software. Next, the cloud service will instruct these gateways to trigger the deployment of the GeneSIS model within the gateway. Fleet assignment does not impact all the edge devices at the same time: The DivEnact tool will re-evaluate the assignment in a pre-defined interval (5 minutes in this case). When new devices join the fleet, previously inactive devices are back online, or the context of some devices changes, the tool will re-evaluate the assignment, and trigger device deployment if needed. The result of the gateway deployment will be sent to the DevOps teams.

As a result of this scenario, we can see that only the devices with Arduino attached are deployed with the new variant, while the other were kept with the previous variant.

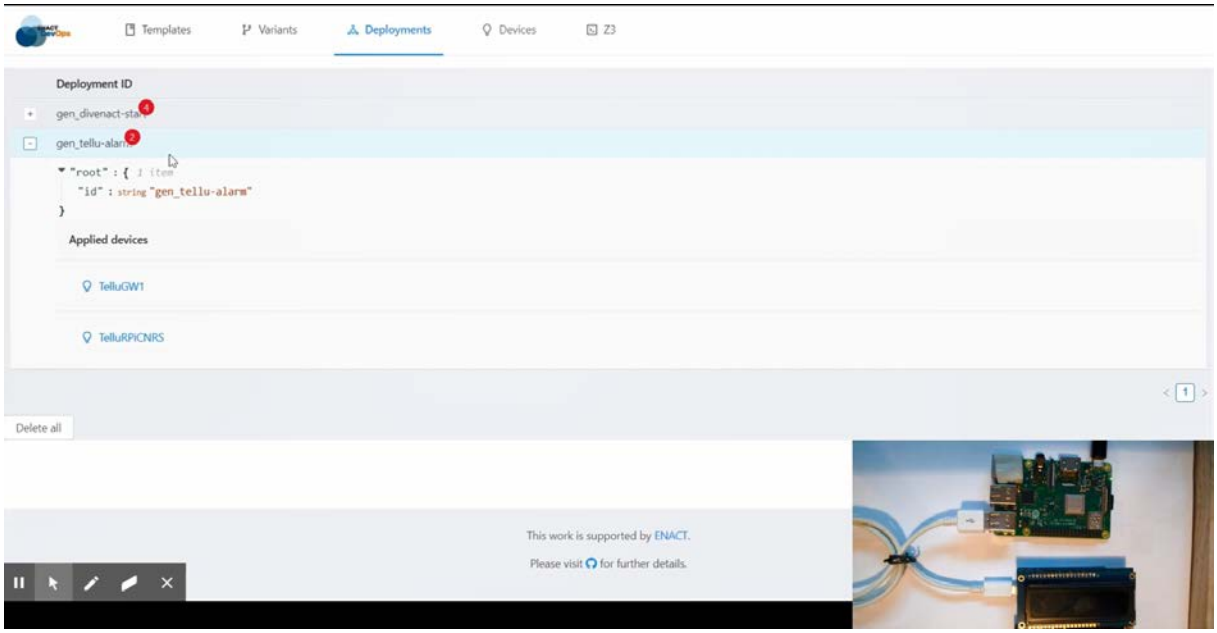

Figure 41. Only the gateways with Arduino are updated

On the other hand, if we unplug the Arduino board from the gateway, the gateway will be automatically recovered to the previous deployment, as shown in the Figure below.

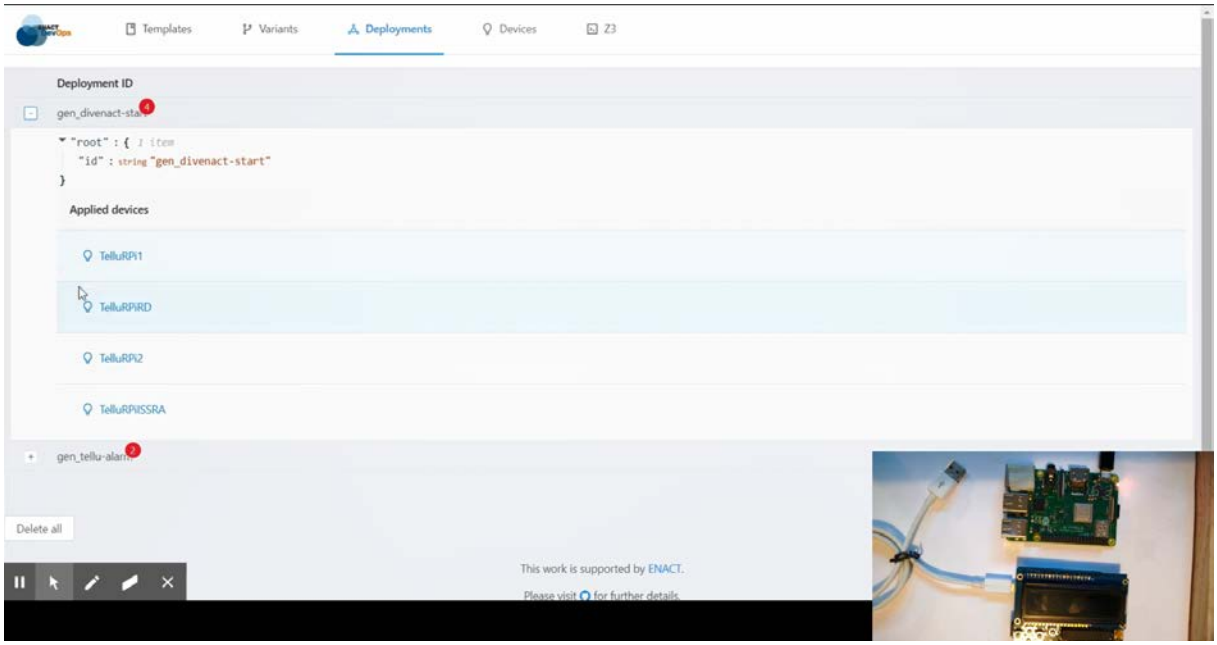

Figure 42. Recovery to the previous deployment

From the end-users' perspective. Two types of end-users are involved in this scenario, i.e., the patients and the nurses.

From the patients' point of view, they will be informed of the new feature, and provide consent to obtain the alarm button devices. When they receive the hardware, they can plug it in to the gateway at any time. When the new software is deployed on the gateway and the connected button's microcontroller, the LCD screen on the button will blink, reminding the patient to go through an enrolment process, i.e., login to their mobile App and type in the code shown on the LCD screen, to verify that the patient him/herself owns and controls the button, as shown in Figure 43, where we received the enrolment code from the LCD attached to the Arduino board, and we need to enter this code to the Evidian service to verify that we are the current owner of the Arduino device. After that, in case of emergent situations, they can press the button to initiate video communication with the supervising nurse.

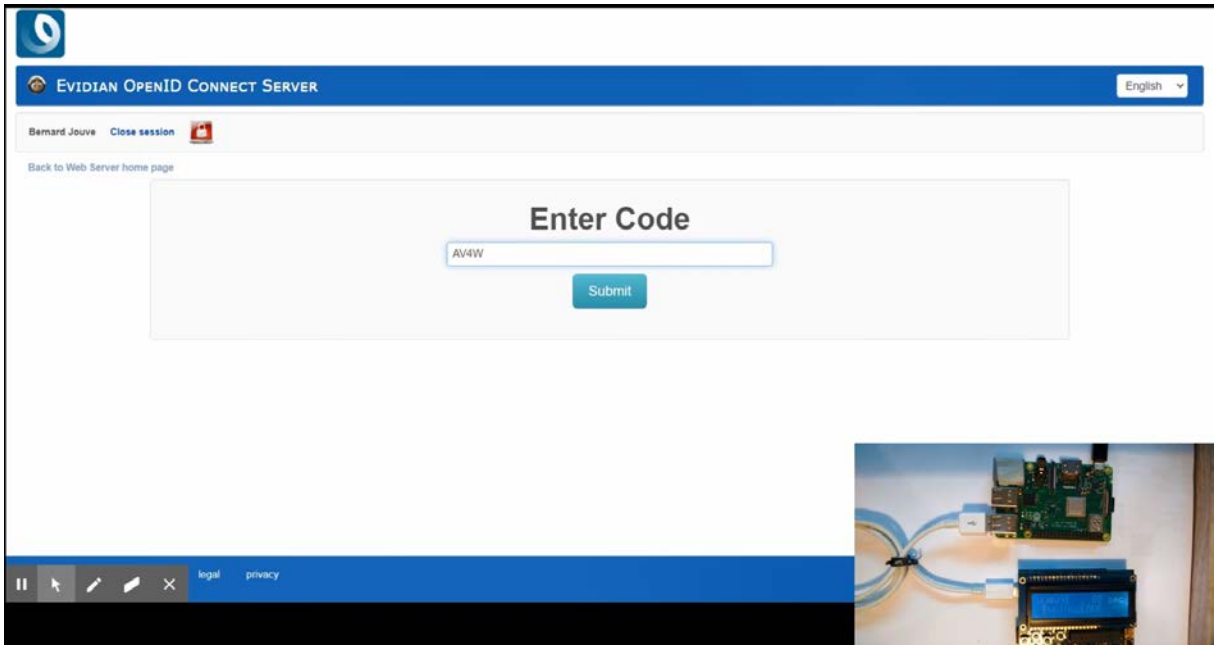

Figure 43. Device enrolment to the CAAC tool

From the nurse's point of view, the only change introduced by the newly assigned and deployed software is that some of them (i.e., the ones supervising patients who have been provisioned with an alarm button) will have access to the camera when a patient pressed the emergency button - a feature that is otherwise restricted in less critical circumstances.

2.2.2.2 The Risk Management Scenario

We tested the Enact Risk Management enabler to cope with the management of risks as an integral part of the Dev process. For this we applied central components of the cloud-based eHealth platform of the digital health use case as target. The overall set of components encountered, and their interactions are depicted in the figure below.

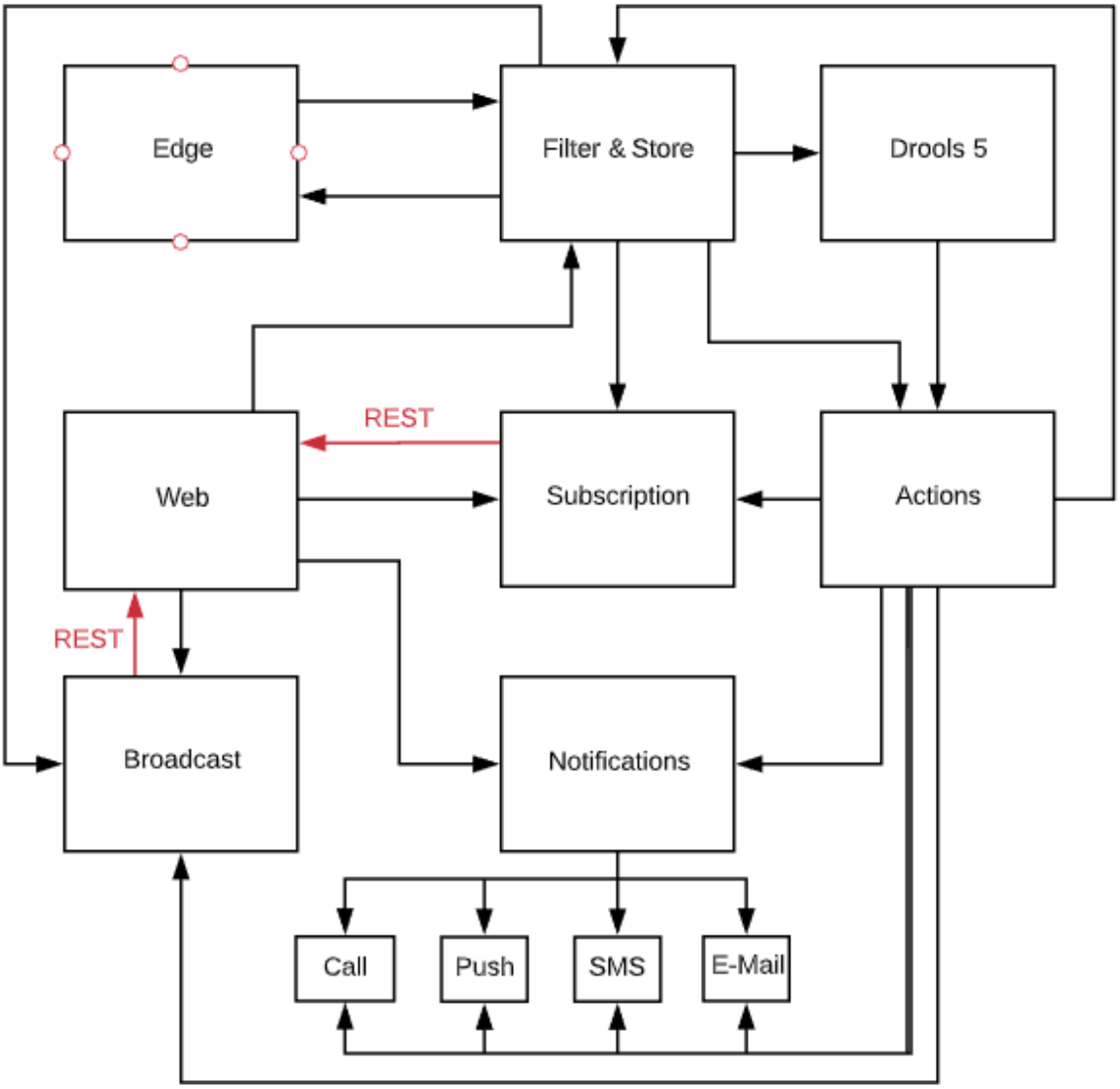

Figure 44: eHealth platform key components and interactions

This part of the eHealth platform enables to build structures that allow to catalogue and store data from IoT-devices, and that can perform actions based on data from these devices.
The Edge represents bidirectional communication with the IoT devices, and the web is an interface for managing and reading data. The Drools component includes logical rules and process the data and determines whether or not any actions should be triggered based on received data and events. These actions can include to create an alarm, notifying a user, create an external event, update a state etc.
For this evaluation and validation scenario of the Risk Management enabler we have focused on the communication channels with the end user. There are in general four channels for communicating with users from the eHealth platform. These are Push Notifications for apps, Short Message Service (SMS), E-Mail and Calls, with SMS and Push Notifications.

Risk Analysis baseline

Currently, Tellu risk management procedures are supported by an Excel based Risk Analysis templated where the main concern that is analysed is Security related to confidentiality, integrity and availability. This is illustrated in the figure below

Figure 45: Current excel based Risk Analysis template

The Scheme is excellent at rating and prioritizing risks, but it is lacking in opportunities to properly track progression and to follow up that issues are being dealt with. Furthermore, the usability is quite poor.

We do assign issues to a developer related to mitigation actions, but this is managed more manually and not properly supported in an integrated toolchain, thus it is a risk to lose trace of the link between the risk, the mitigation actions and the DevOps process with development testing, deployment operation etc. Moreover there is a risk that mitigation actions are missed out as well as a risk of inconsistency because the spreadsheet includes links to Jira issues and deadline dates. These may be updated independently while there is a need to update both places to keep the consistency.

Also, versioning is a significant problem as there is no automatic support and there is no tracking of the changes between two versions of a risk analysis file. These are main issues that we liked to see whether the ENACT Risk Management enabler could improve on integrating with popular Issue management systems like Jira.

Evaluation and validation

The evaluation and validation scenario described here is related to the SMS component and a set of issues that have appeared from the Risk Analysis related to this component. This includes integrity and availability risks such as:

- Incorrect configurations being pushed to the component
- Third party SMS Provider is down, or is not responding
- A Customers daily or monthly SMS threshold limit is exceeded

Applying the Risk Management enabler, we first modelled the system context which in general includes all the main components described earlier for this part of the eHealth platform.

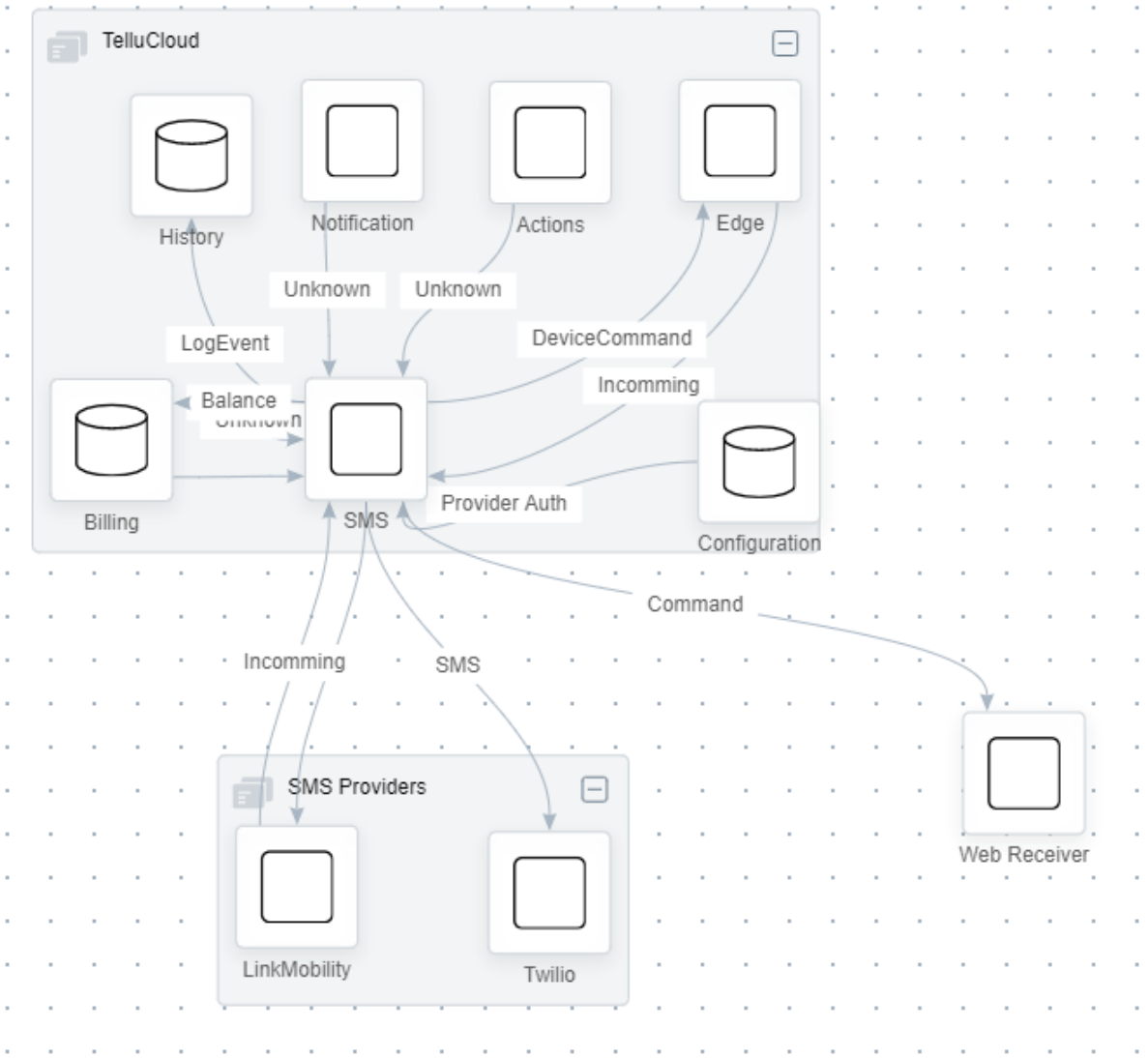

Figure 46: SMS Application View

The model includes the SMS component, and shows its dependencies and interactions. In general, the ENACT Risk Management enabler supports to analyse risks per component.

For the SMS component one risk encountered is related to a vulnerability that the configuration is corrupted and still the configuration is deployed to the SMS component. This may for instance lead to a state where the SMS component is not ready to send SMS's at all.

We apply the tool to define this Vulnerability and connect it to the service, and we define the risk related to this vulnerability.

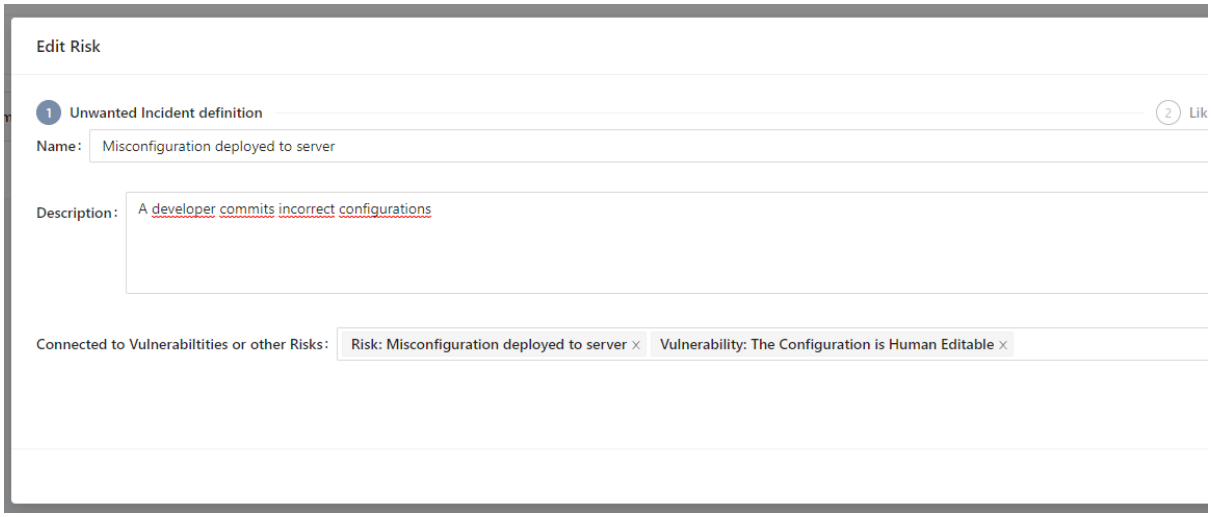

Figure 47: Risk Editor Step 1

The further process is to perform further risk analysis related to this risk in terms of *Likelihood* and *Impact*.

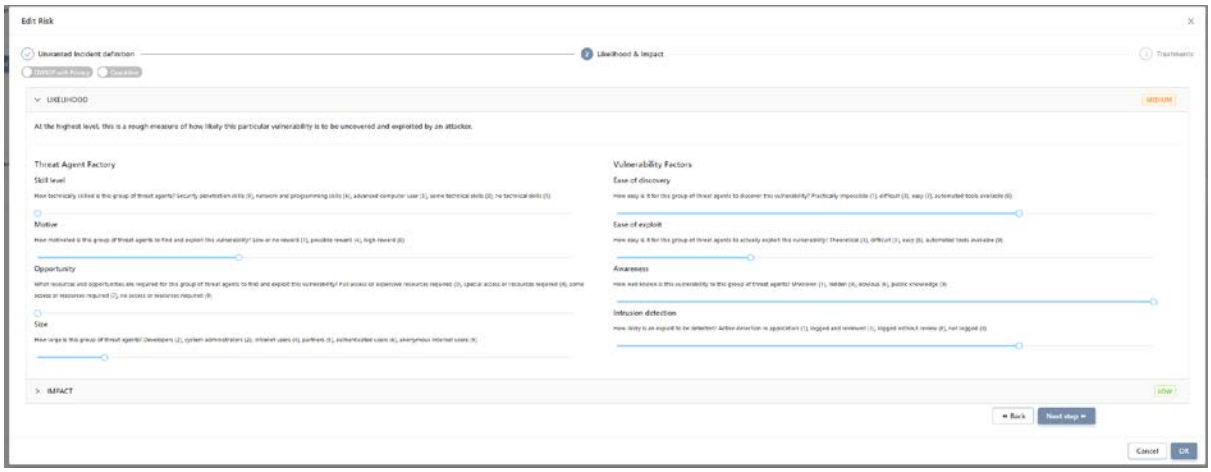

Figure 48: Risk Editor Likelihood

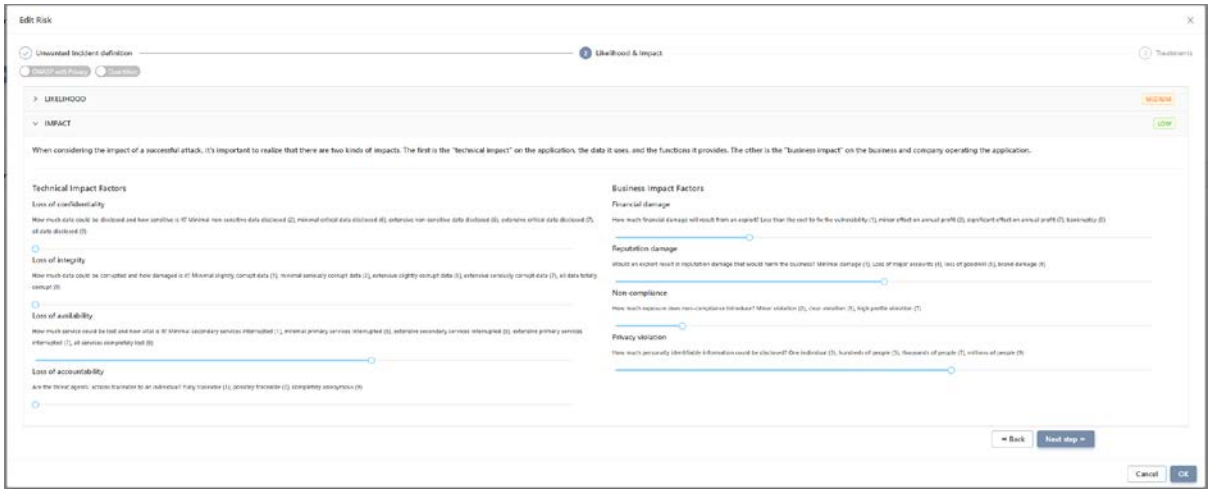

Figure 49: Risk Editor Impact

The final step prescribed for the risk analysis process in the ENACT Risk Management tool is the Treatments. For this risk, a particular treatment is to improve the process of deploying configurations.

We now have defined the risk and done the risk analysis for the SMS configuration and we have it prioritised similar to what we have in our existing Risk analysis system.

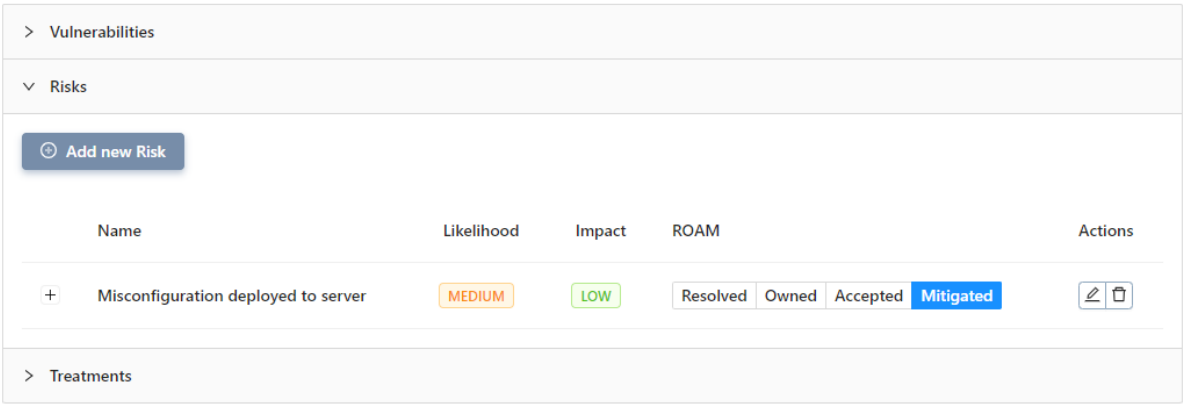

Figure 50: List of Risks for Configuration

Another problem discovered in the risk analysis related to the SMS component is that the mechanism that is meant to prevent the SMS system to be able to send a lot of SMS due to some errors or misuse can actually prevent the system or user to be able to send SMS's that they should deliver.
As this is a process related risk, we model it in the Data Flow Diagram provided by the ENACT Risk Management enabler (see the figure below).

The risk is related to the threshold mechanisms allowing the customer to set a maximum limit for the number of SMS's sent per day and per month. Obviously, it may be a risk that important SMS related to an alarm or similar can be stopped when this threshold is reached.

We register this Risk/vulnerability in the ENACT risk management tool related to the process *Check Limits.*

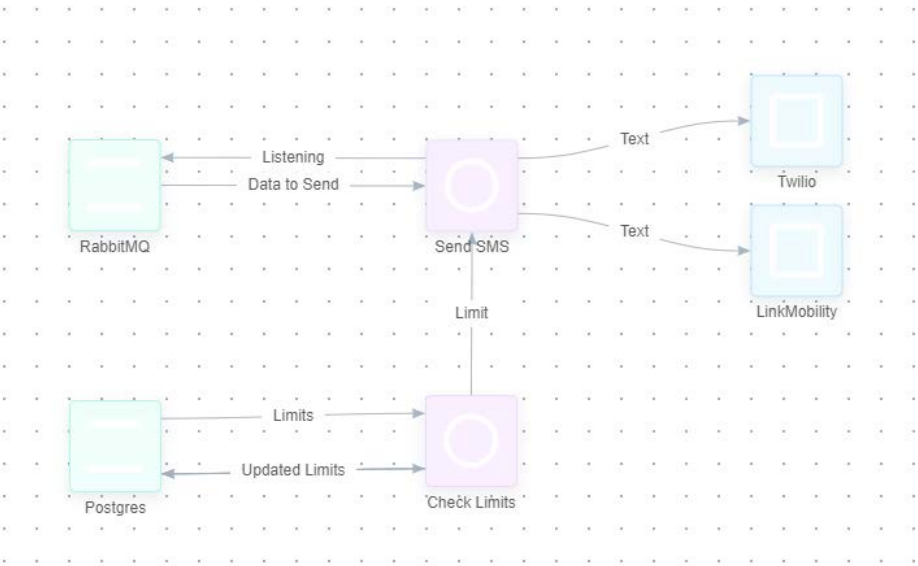

Figure 51: Data Flow Diagram for SMS

When modelling with the Data Flow Diagram, we have the option of using entities Datastore, Process or Entity. It would have been nice if it was possible to use entities already defined by the Application View, however this is not currently supported in the tool, thus, there will be duplicates in the Asset Lists.

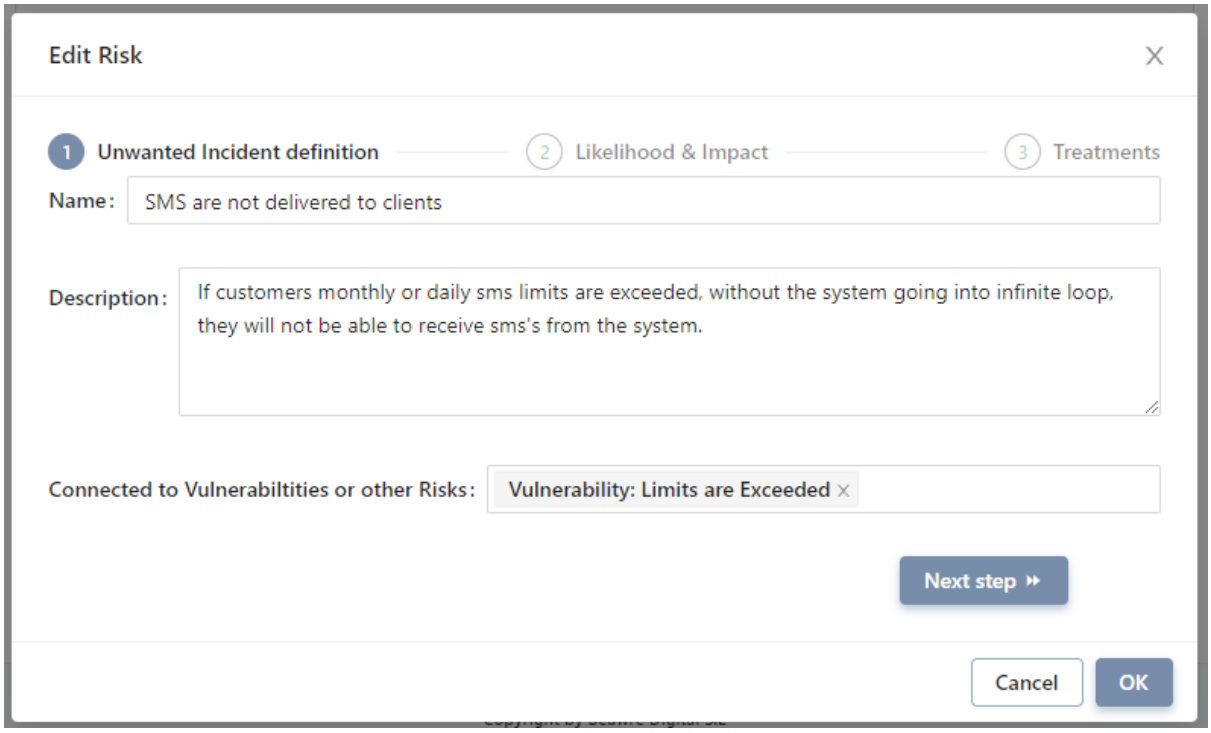

Figure 52: Risk Editor

The treatment for this use case is to have an hourly database check, whether the thresholds have been surpassed, and notify relevant stakeholders if they have, so that this can be checked, and followed up for example to increase the threshold for the day or the month.

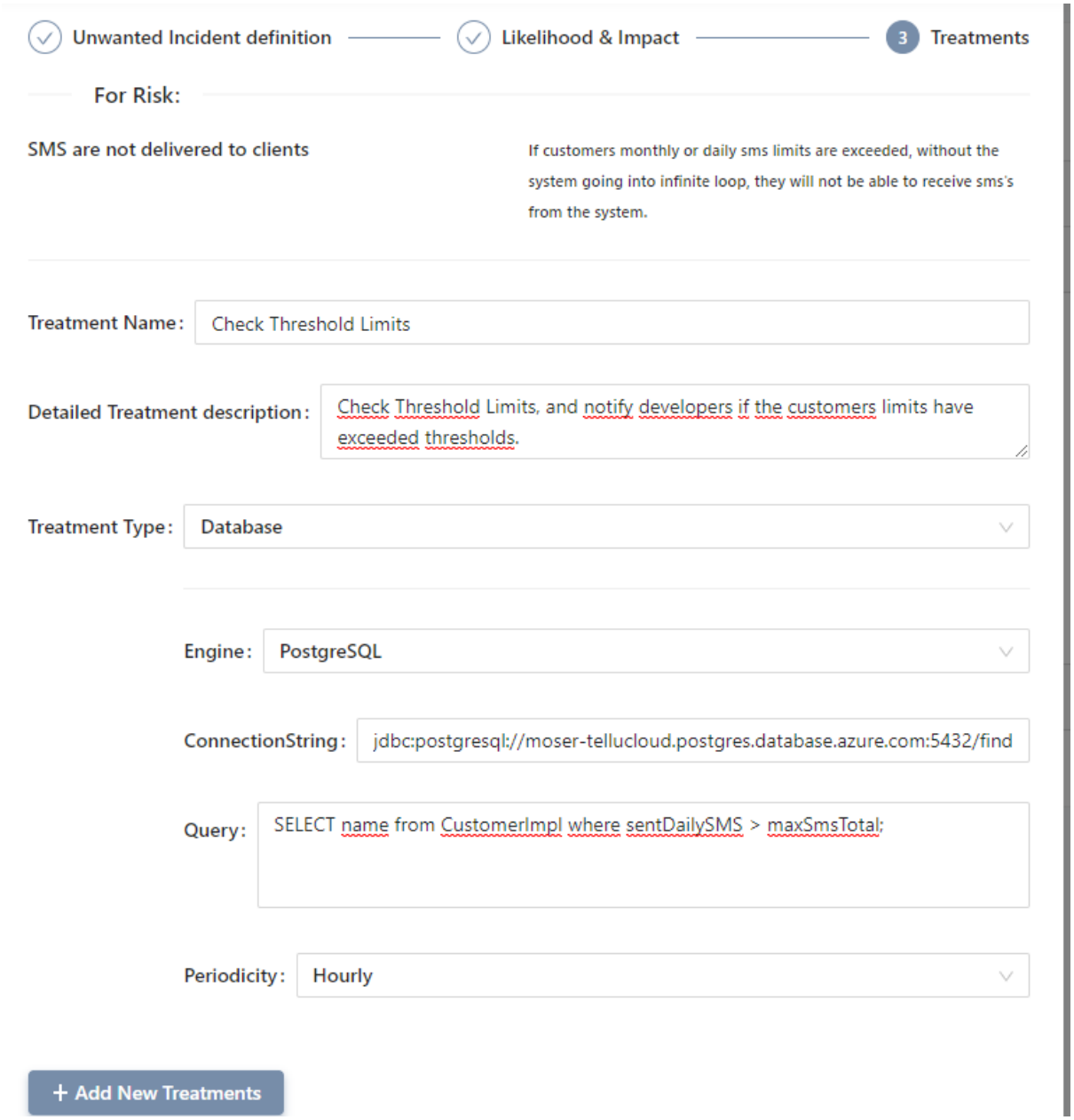

Figure 53: Risk Editor Treatments

In the Enact Risk Management enabler Treatments may be connected to Repositories or Issue Managing tools such as Jira. This is an important feature in order to have risk management as an integral part of the DevOps tool chain. In the tool this coupling is easily set by linking to Jira issues as illustrated in the figure below.

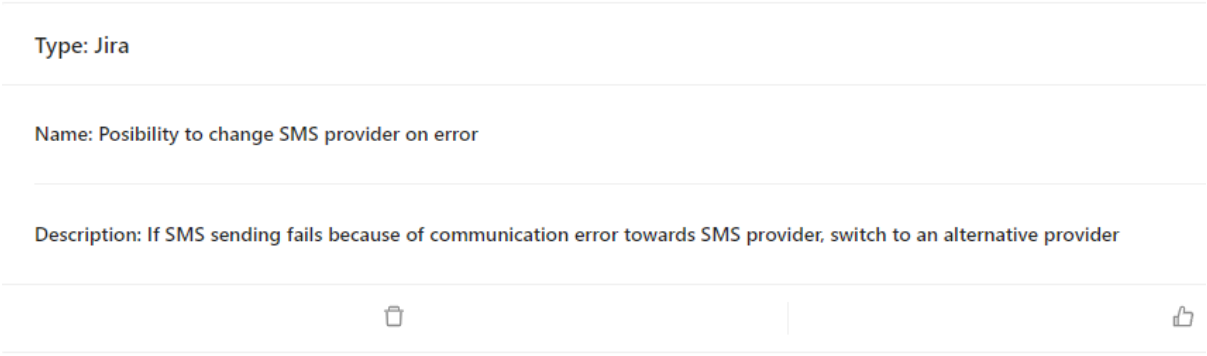

Figure 54: Jira Treatment

As opposed to do manual configuration of Risks and vulnerabilities, the Enact Risk management tool provides functionality to automatically generate vulnerabilities by automatic analysis on the models and by replying to some questionnaire schemas provided in the tool. From these, also potential risks and mitigations are proposed. For example, for the model of the SMS component and its relation to the Postgres database and by filling out a questionnaire provided by the expert system of the tool (illustrated in the figure below) the tool will automatically derive potential vulnerabilities and related risks and mitigations.

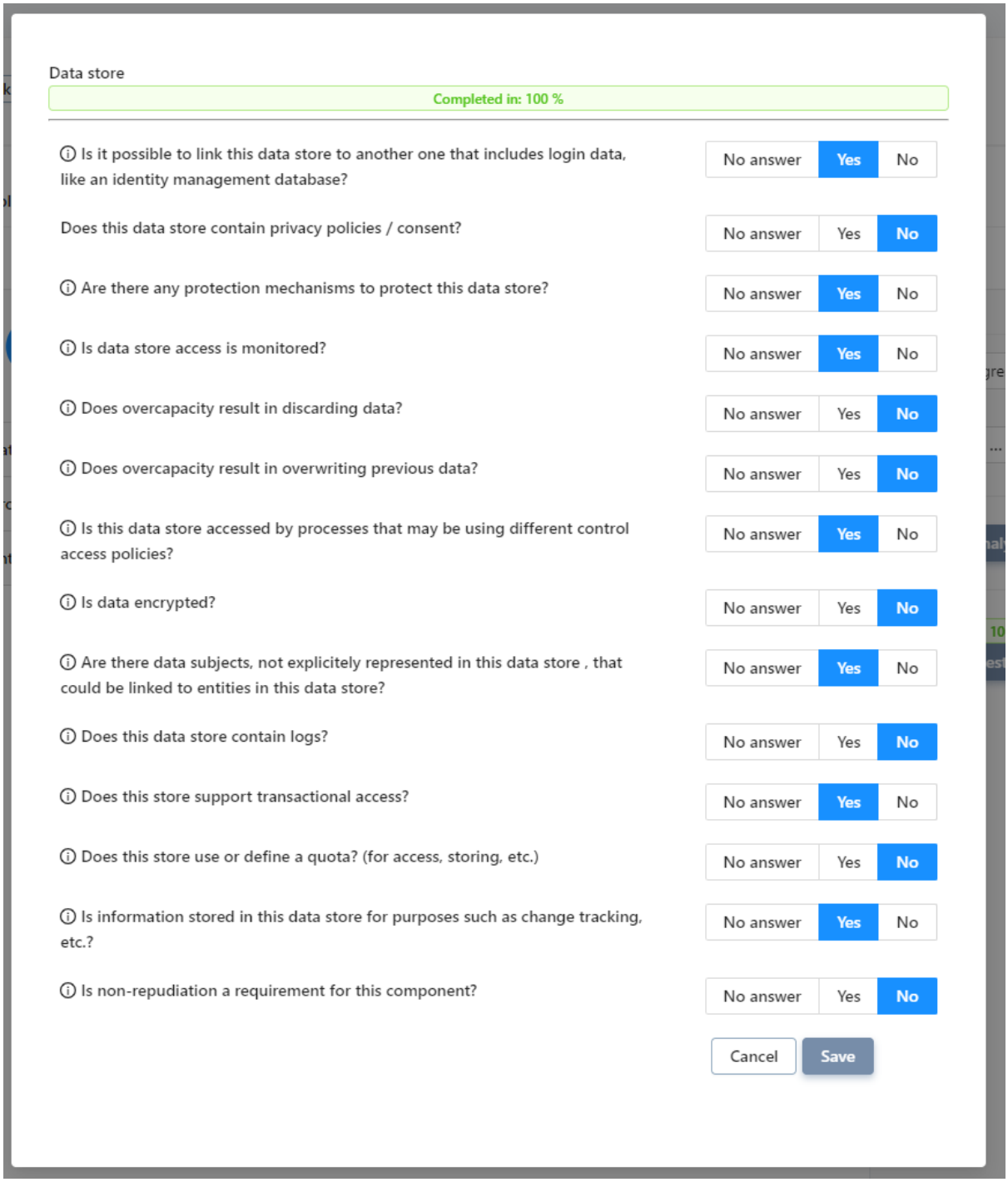

Figure 55: Risk Management Questionnaire: Example of questionnaire for DFD element used for automatic vulnerability detector

Applying the Automated Vulnerability Detector feature of the tool, it will generate vulnerabilities for this part of the model. For example, in this evaluation scenario applying the SMS component and its relation to the Postgres database coupled with the questionnaire, we got generated 19 Vulnerabilities resulting in 82 Risks and 133 Treatments. Most of these are redundant to what we have encountered in the risk analysis or the proposed vulnerabilities are not relevant, still some of them are issues and vulnerabilities that where not detected in the risk analysis. One example of relevant risk that was detected by the tool was, “Activity Hijack”, related to vulnerability applying implicit intent in inter application communication which is an important concern to handle in our SMS service, and the tool also then propose mitigations to handle this risk. This is illustrated in the screenshot below. Thus, in this

evaluation case the Enact Risk management tool successfully detected vulnerabilities and risks that was not encountered and ensured a better coverage of the for the risk analysis.

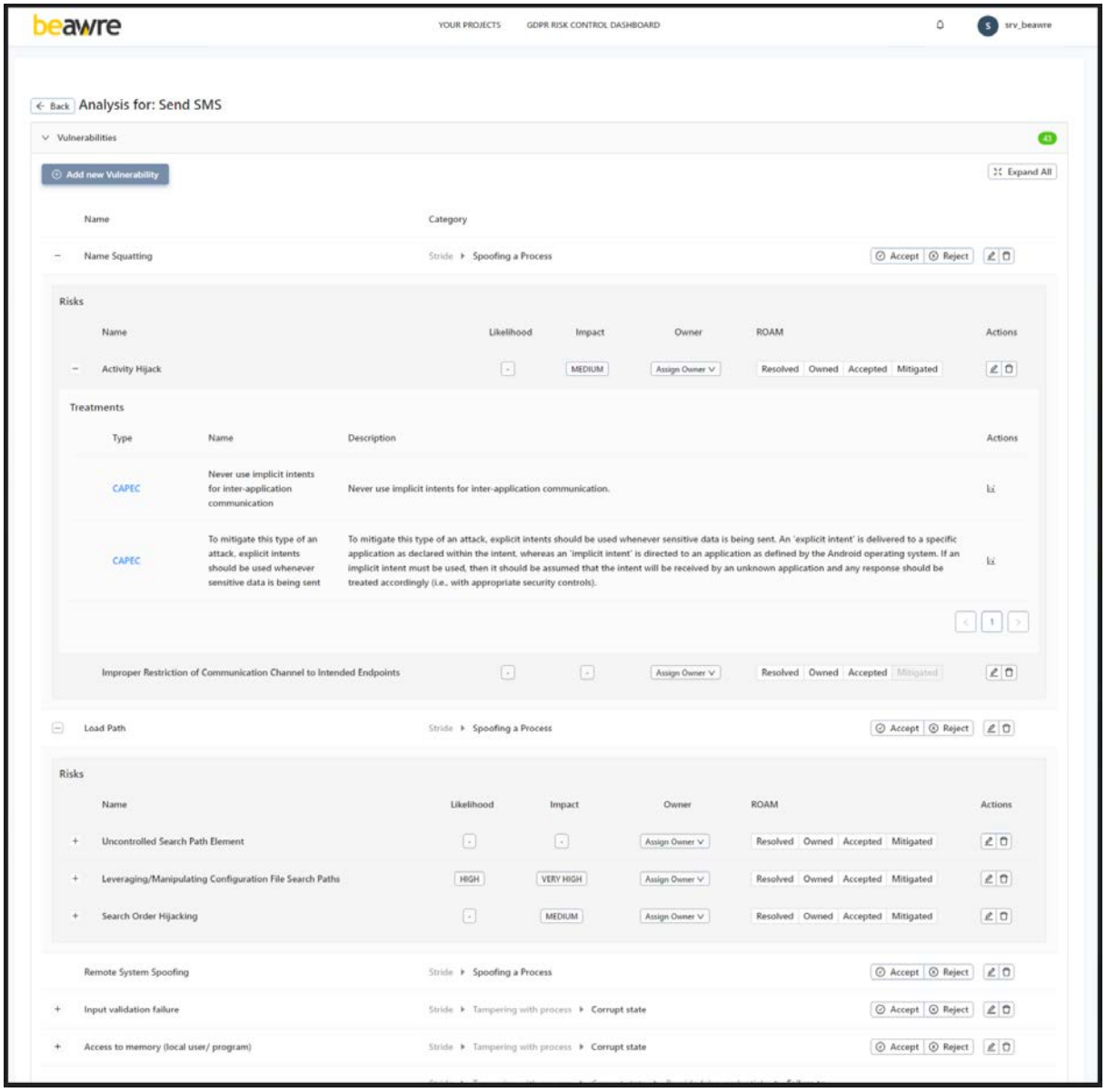

Figure 56: Risk management tool;Vulnerability, risk and mitigation generation : Tree of detected vulnerabilities, risks and mitigation actions proposed against a single component in the Risk Management tool

Evaluation summary

In general, our evaluation is that the tool appears solid and include significant features that is helpful and relevant to support risk management as part of the DevOps process. In particular it I closely coupled to the system design and architecture models as well as the component interaction design. Thus, in general you provide the same or similar models as you have specified as part of the system design activities. Moreover, it supports the linking and integration with popular issue management systems such as Jira and it provides quite advanced features in terms of supporting risk coverage by providing an expert system provides relevant questionnaires based on the provided system models and automatically provides (through generation) proposal of vulnerabilities, risks and treatments.
The evaluation and validation scenario illustrates that we are able to create detailed risk analysis for our system and that the tool support a systematic approach of encounter risk component by component.

In terms of usability the ENACT Risk Management enabler is still a bit immature and not always intuitive and sometimes it appears a bit complex. For example, the approach of a risk model per component, and in addition add a flow chart, there can eventually be a lot of component level risks to keep track of and the intersection with the DataFlow risk is not always easy to manage.

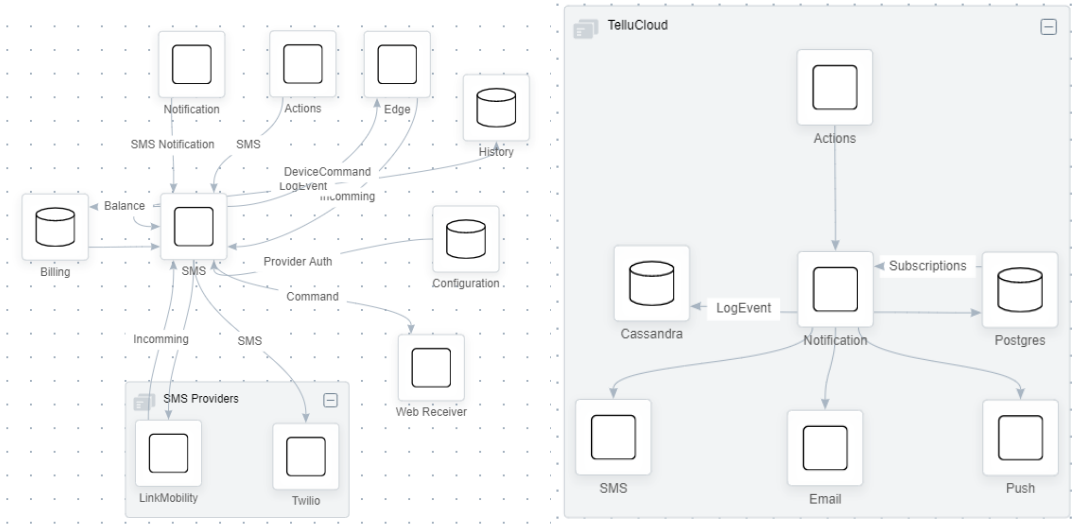

Figure 57: Illustration of view interaction management : Showcase of the architectural components in the TellU cloud

Also, the support for providing the overall application risk picture can be improved. For example, you typically end up with duplicate components as illustrated in the figure below. This may be appropriate to get risk analysis in context, but it gradually makes the overall risk picture situation a bit confusing.

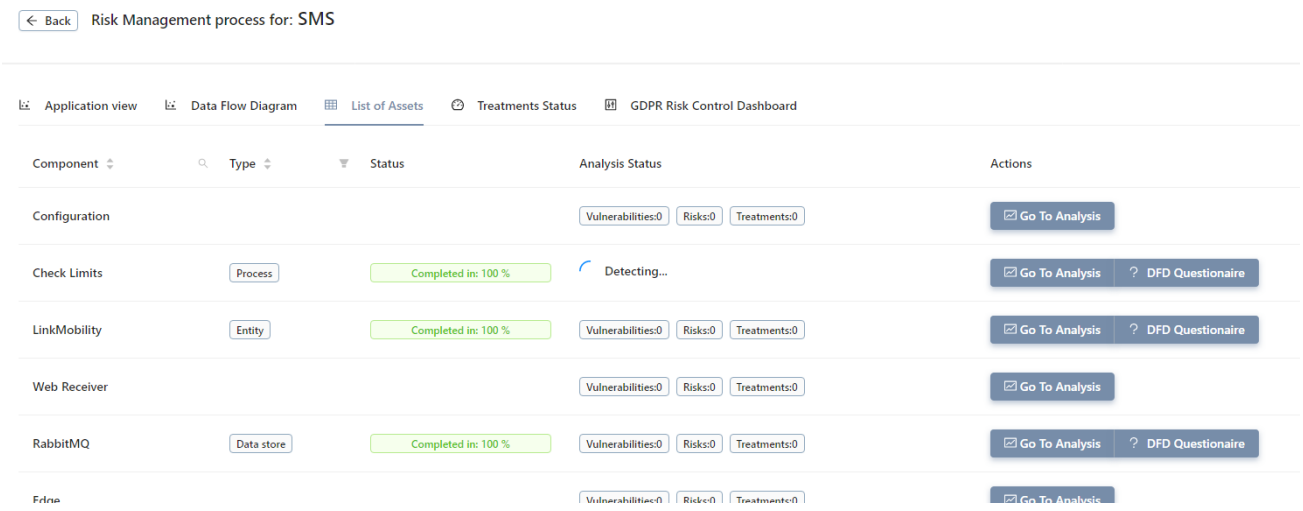

Figure 58: Illustration of Duplicate components problem: Dashboard presenting the overall status of the risk management process

For the overall risk picture, we also end up with a long asset list where several components have the same name, and it is difficult to distinguish them from each other.

The support for specifying and managing the treatment we believe is very good and in particular that we as mention can have direct link to our issue tracking system jira is a clear improvement for us.

We also encountered a set of bugs and errors during our evaluation and validation that was reported including errors such as:

- Some data was not stored
- Some data was stored several times implying creation of unwanted duplicates
- Some invalid input caused components to become corrupt
- In some cases, data were is not loaded correctly, or was mixed with data that was already in memory etc.

These are obviously things that will are going to be fixed as part of the ongoing development of the tool, however, these errors and bugs had some impact for some parts of our evaluation and validation. We are still in general impressed about the progress and evolution we have seen for the Enact Risk Management tool during the course of the ENACT project and se definitely see significant andvances this tool represents related to state of the art.

2.2.2.3 Test and simulation scenario

The Digital Health use case has been applied to evaluate the ENACT Test and Simulation (T&S) tool. For the development and evaluation of tool according to various requirements the Digital Health use case was applied. The overall architecture of the set up for the use case based evaluation is depicted in Figure 59. The Data Recorder records the data from a production gateway and stores the recorded data into the Data Storage, those recorded data are used as testing datasets for non-regression testing. The simulated sensors simulate some sensors in the gateway and send data to the CloudAgent via Internal Broker. The simulated actuators generate actuated data to control the actuators in Arduino, the actuated data sent via Sensor Bridge. The Regular and Malicious Data Generator generates dataset to be used as input in different testing scenarios such as: the scalability of the message queue, the resilience of the system with abnormal behaviour of the sensors, and the possibility to handle some cyber-security attack (DoS attack).

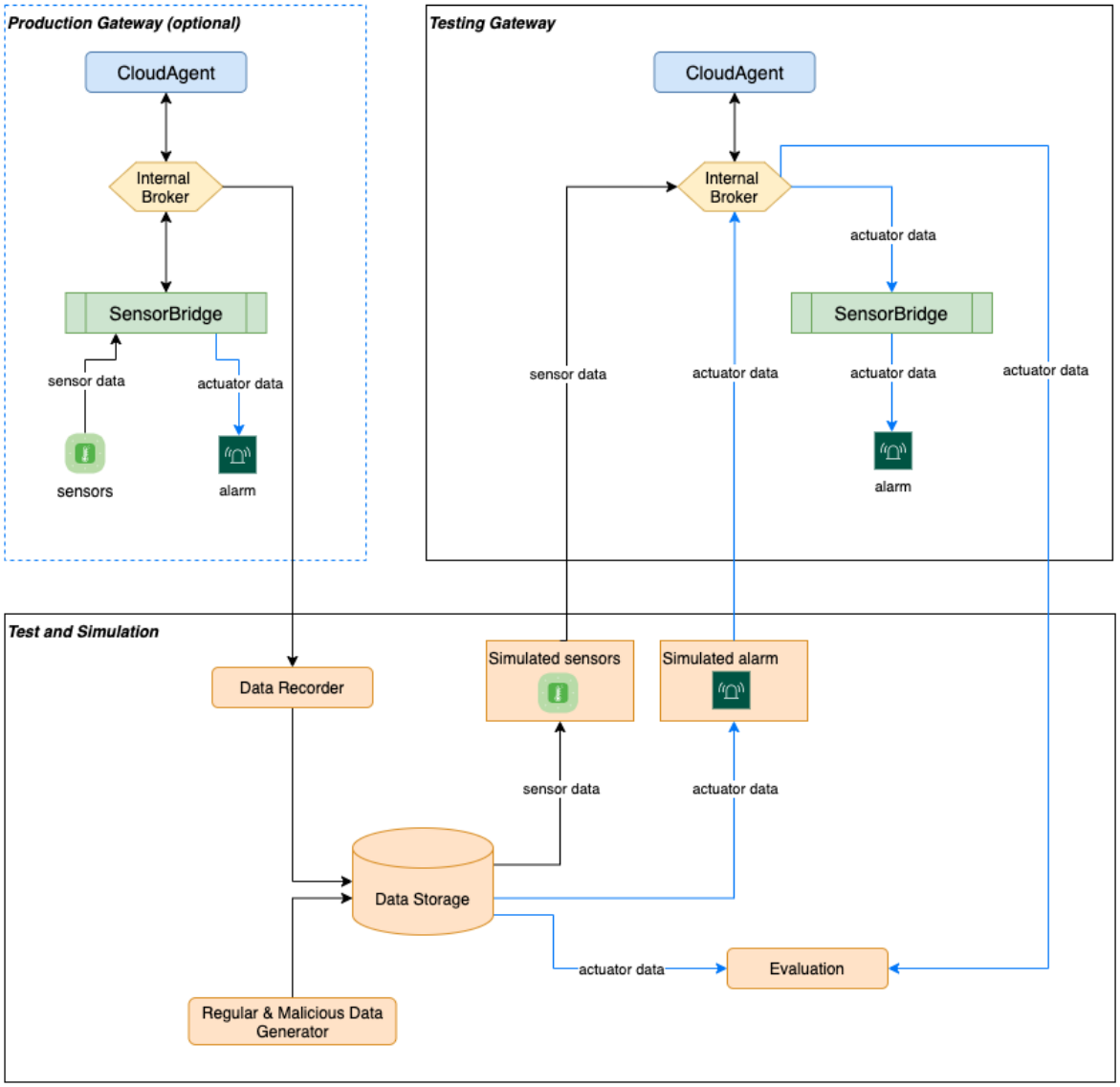

Figure 59: Test and Simulation in eHealth use case

Evaluation and Validation

The Test and Simulation enabler has been evaluated in Digital eHealth use case to validate a scenario with the following steps:

- Simulate a single safety alarm
- Simulate 100 safety alarms to test the scalability of the eHealth gateway
- Simulate the different behaviours of safety alarm

The Figure 60 illustrates the simulation of a TellU device. The device contains a single Safety Alarm as a sensor because this safety alarm sends data to the Gateway as all sensors. The device connects to the testing gateway which locates in an Azure IoT Hub, so the device needs to establish a connection type of AZURE_IOT_DEVICE. In simulating mode, there is only one instance of this device is created because the scale factor is 1.

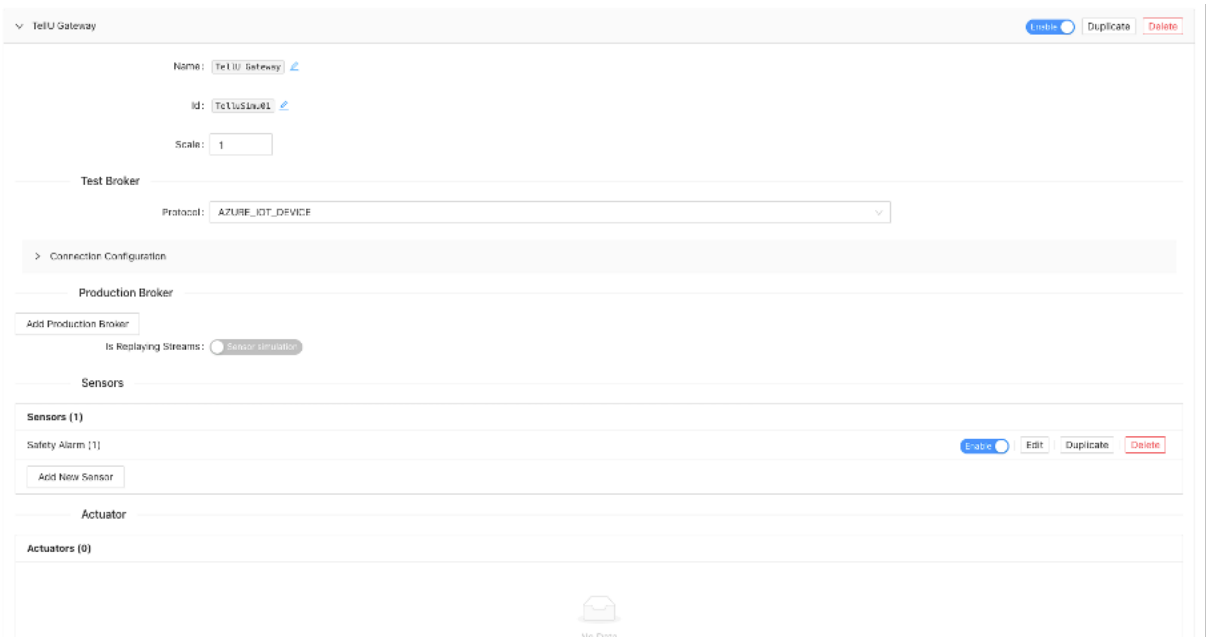

Figure 60: Simulation of a TellU device

The Figure 61 presents the configuration detail of the simulated safety alarm. The safety alarm has been configured to have the data source from Regular & Malicious Data Generator which means that the signal of the safety alarm will be generated while simulating, and the value of generated data message will be based on the configured behaviours. The safety alarm has two data fields, the id and the button. The data published has the JSON format as showed in Figure 62. The safety alarm has been configured to have normal sensor behaviour and it generates a data message in every 5 seconds.

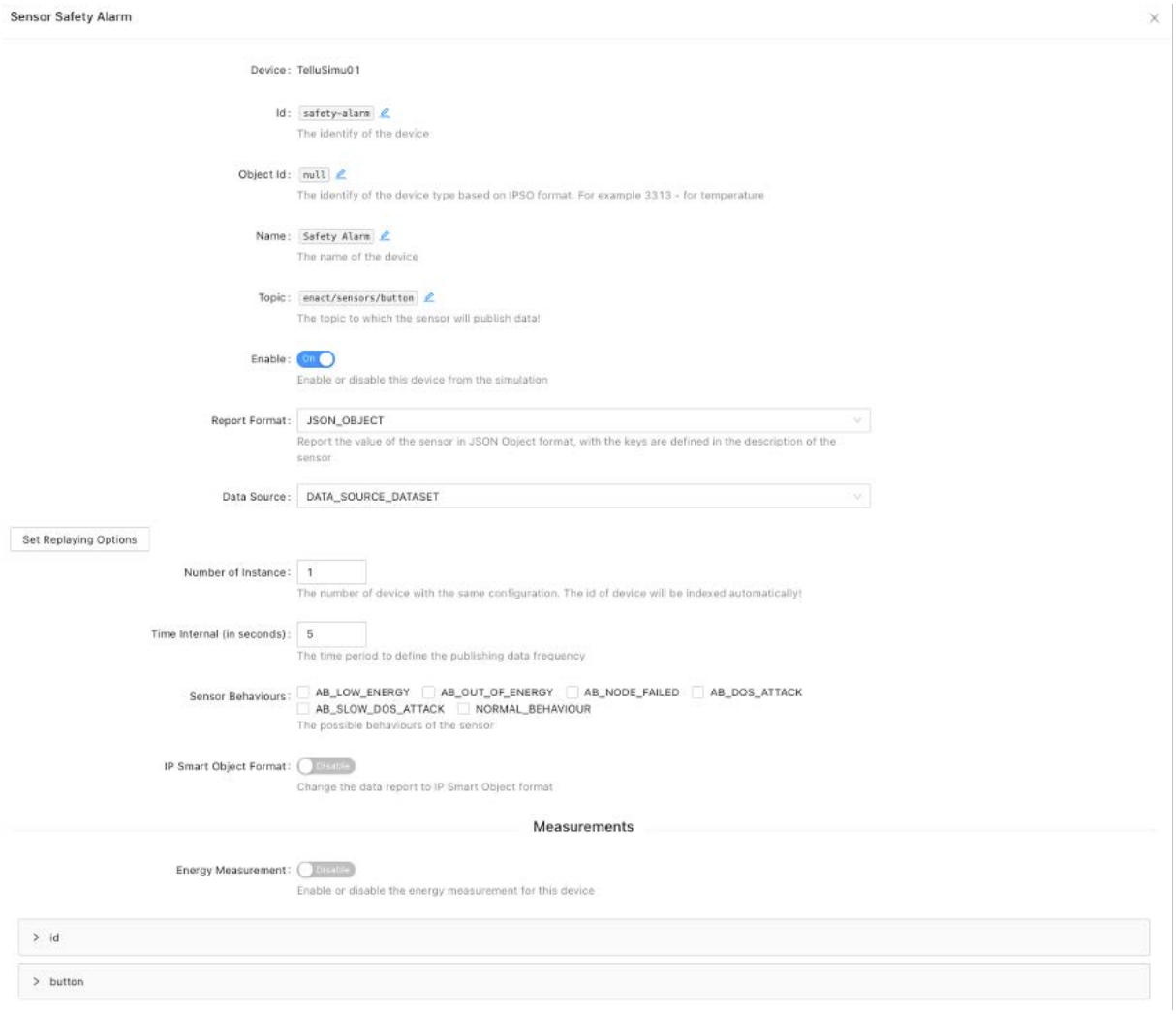

Figure 61: Simulated Safety Alarm configuration

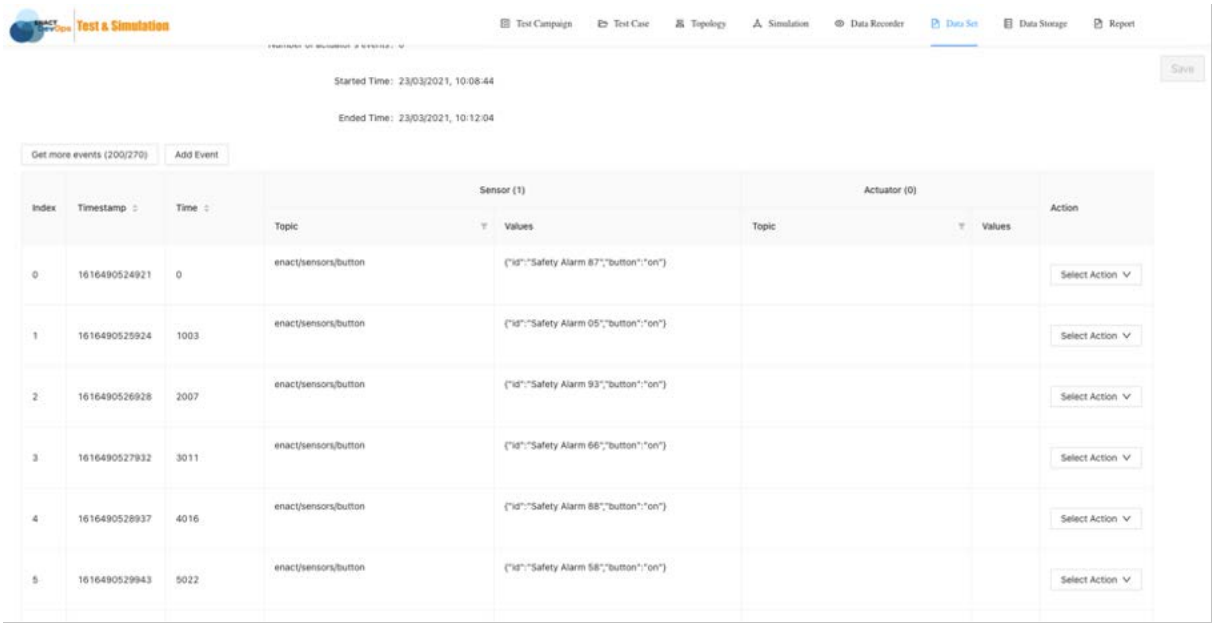

Figure 62: Generated dataset for a TellU device

The safety alarm sends a data message contains the alarm status along with the id to identify the patient who triggers the alarm. The Figure 63 show the possible value of the alarm status: ON/OFF and the possibility to simulate the different behaviour of this value, such as: AB_FIX_VALUE - a Safety Alarm sends a fix value (always ON or OFF), AB_INVALID_VALUE – a Safety Alarm sends an invalid value which is not ON nor OFF, this is the case when we can simulate a cyber security attack scenario where a safety alarm has been hacked and sends invalid data to crash the TellU gateway (if the TellU gateway does not support to handle the invalid data).

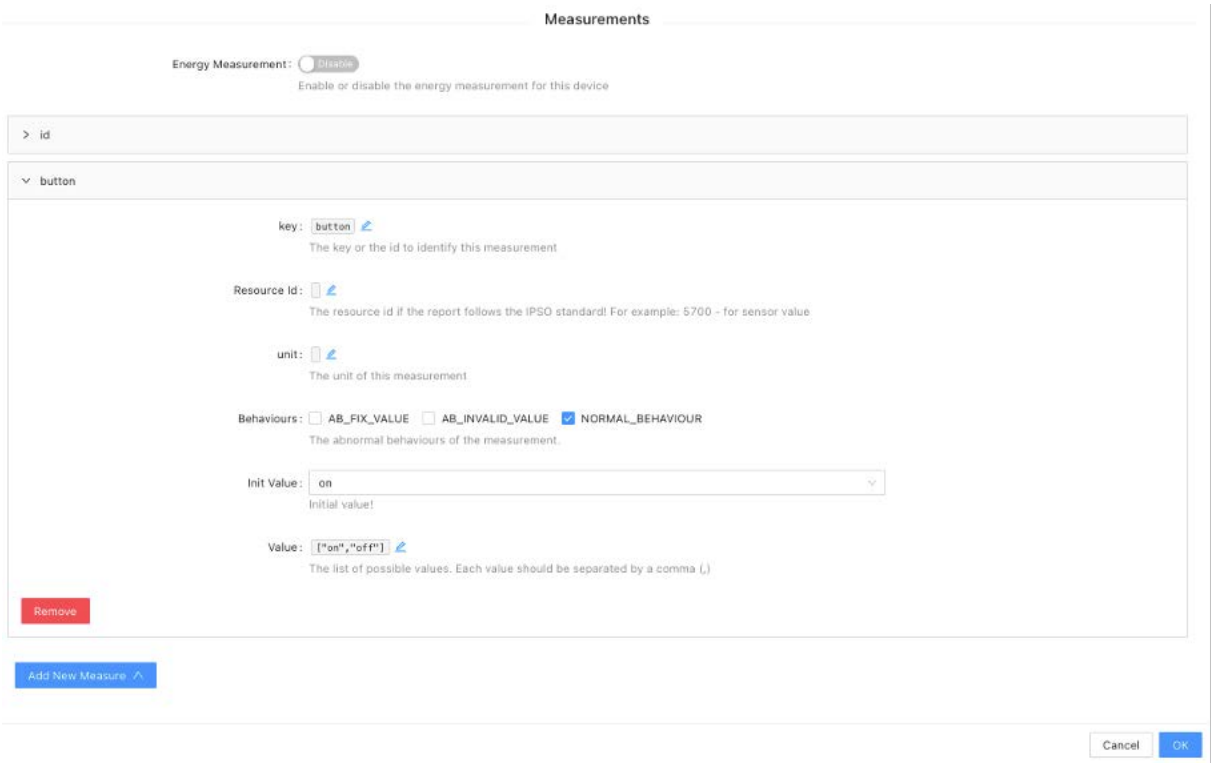

Figure 63: Safety Alarm status values

In collaboration with SINTEF and Tellu, we visualise the simulation process by using PowerBI to build a dashboard which presents the safety alarm data published by the Test and Simulation tool. In the dashboard, the bar chart shows the number of times the safety alarm has been triggered on each patient. The Figure 64 is the status of the PowerBI dashboard before simulating 100 TellU devices. As be seen in the chart, there are only 7 safety alarms have been triggered, and each one has been triggered one time.

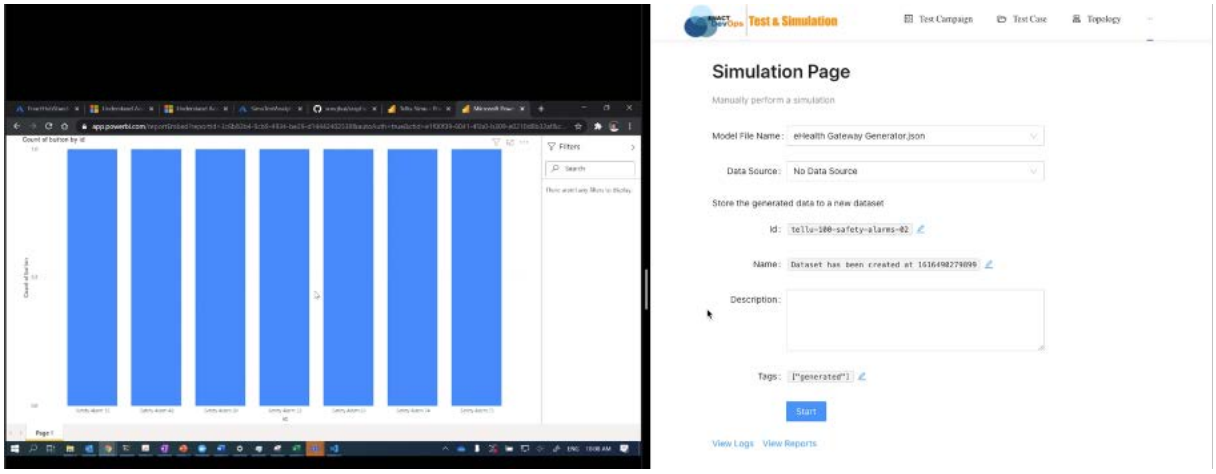

Figure 64: The PowerBI dashboard before simulating 100 TellU devices

The Figure 65 below shows the PowerBI dashboard while simulating 100 TellU devices. The number of safety alarm is increased and there is different in the number of times each safety alarm has been triggered.

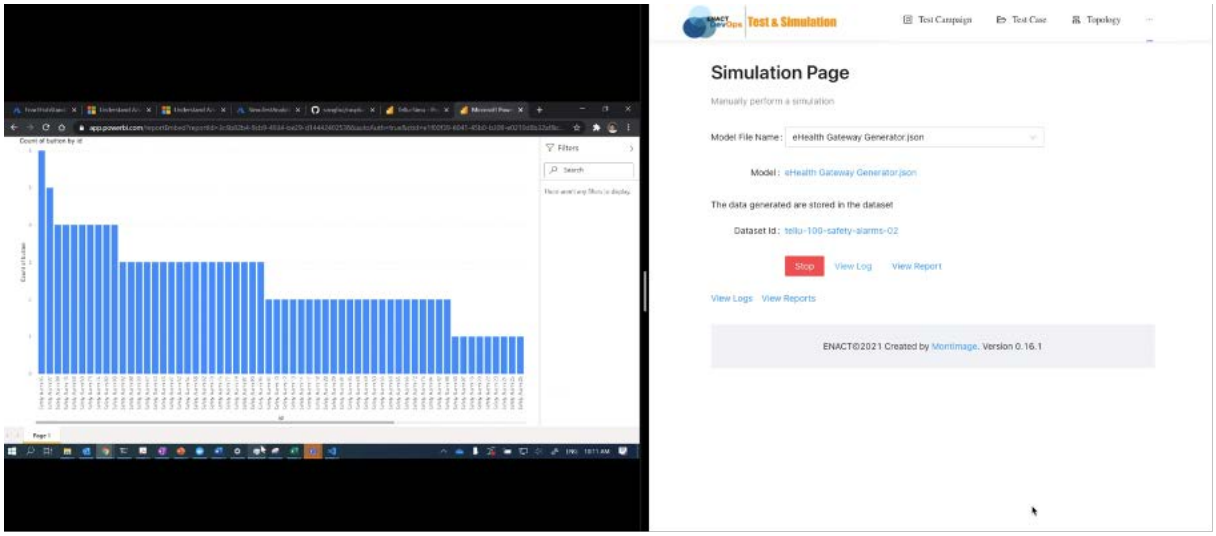

Figure 65: The PowerBI dashboard while simulating 100 TellU devices

The Figure 66 shows a different monitoring view where we can see the number of in/out events.

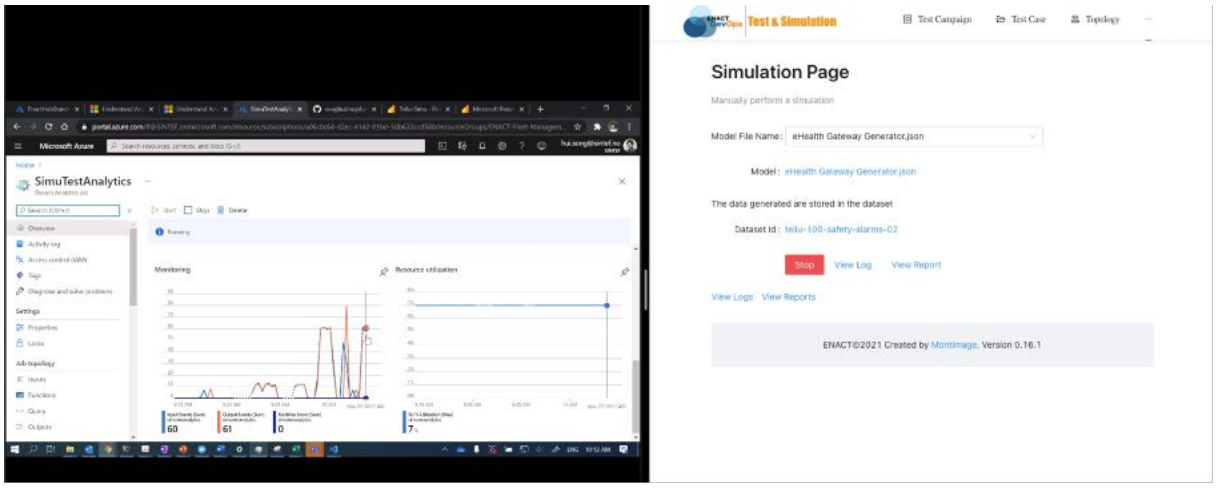

Figure 66: Monitoring status

On the PowerBI dashboard, there is a possibility to see the detail of the data published by Test and Simulation enabler as shown on the Figure 67.

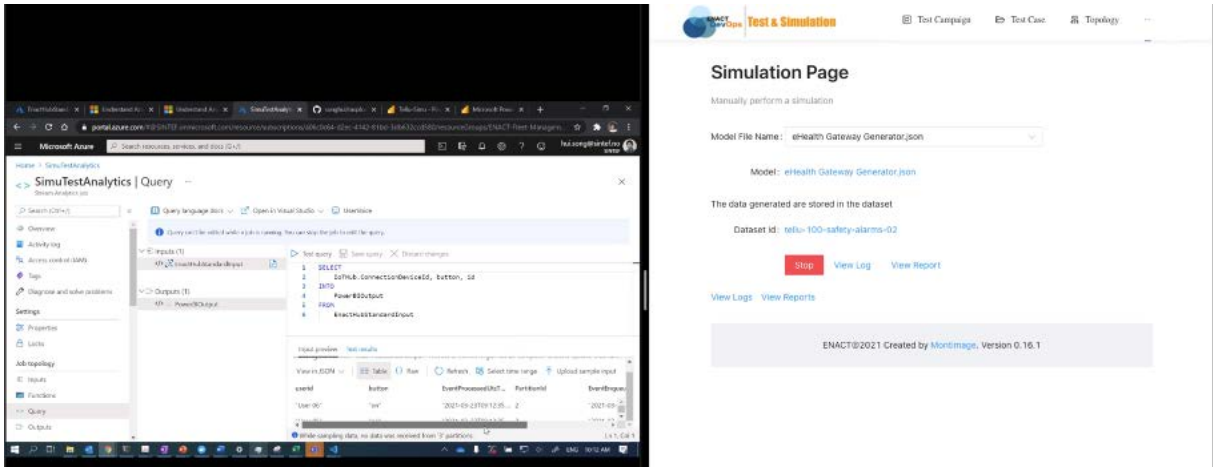

Figure 67: Simulation data signal received by Gateway

Evaluation summary

The Digital eHealth use case has validated that the Test and Simulation enabler can help testing a large set of testing scenario which are not easy to produce in a real environment, from non-regression testing for normal behaviour to scalability testing, abnormal behaviours testing and some cyber security scenarios.

2.2.3 Evaluation and Validation of the Business Case

Business KPI	Comment
Main KPI	
The personal gateway is applied in production The use case has been commercialized during ENACT with hundreds of patients. Tellu hase expanded its business and experienced significant growth in the eHealth domain during the period of ENACT	The use case itself has evolved from being a prototype development to being at production ready system recently sold to customers and already deployed for remote supervision of several hundred patients suffering from the following chronical diseases: COPD, Kidney and Diabetes. Moreover, it has been applied for remote supervision and following up of Corona patients in several municipalities in Norway. The application has got attention from the World Health Organisation (WHO) and will be presented at a WHO conference this spring by one of our customers (a Norwegian municipality). This again leads to attention from Norwegian media houses putting this on their news (including TV news). The support and exploration we have been able to do in ENACT has been significant to reach this success, in particular, the Personal Health Gateway, which is a core component managing the IoT and edge part of our service, the ENACT project has been significant for its development. During ENACT TellU has experienced significant growth, both related to exploiting the ENACT use case to build a production ready remote patient monitoring service and by evolving its already existing remote supervision services in the welfare domain (camera-based and sensor based supervision), which evolution has also been explored as part of ENACT. In figures TellU has grown its revenue 10 times during the ENACT project period (from 0.7 M€ in 2017 to

		7 M€ in 2020) and has grown from 5 employees in 2017 to about 40 to date
Sub-KPIs		
Fasaster delivery and scalability	Time to fix security and privacy issues reduced by 50%	Applying DevOps process for the complete software development lifecycle also for the IoT and edge part makes the lead time for fixing significantly lower. This is in particular due to the Enact deployment bundle (where ThingML is already exploited in the production tool chain in the company), but also the ENACT risk management enabler that supports in detecting security and privacy issues, The CAAC enabler that support dynamic adaptation of access enabling to fix some security and privacy issues on the fly, and the Test and simulation enabler that reduce the time for testing supports this KPI.
	Faster development and more efficient integration, estimated improvement 3x faster dev and 5x lower effort for integrating	Applying DevOps process for the complete software development lifecycle also for the IoT and edge part makes the development faster and integration tasks more efficient. For development, the ThingML enabler play a significant role to reduce development time of low level code required for many IoT and edge devices. ThingML provide proper abstraction and derivation/compilation to running low level code in an efficient and error prone way. Furthermore, the ThingML significantly reduce the time to integrate new IoT and edge devices. The ENACT risk management enabler also leads to faster and more efficient processes in terms of risk management tasks related to the development and integrations and the Test and simulation enabler which reduce the time for setting up and perform testing
	Deployment in production in one day on remote site	An ENACT evaluation underpinning some of the above KPIs is that we were able to provide a production ready deployment of the Digital Health system for ISRAA. Due to the pandemic this was also performed only through online support from Tellu staff
	Fleet management: • Short term: new funding • Current: Human effort required to deliver gateway to patients reduced by 30% • Long term: check KPIs/numbers from FLEET	Through ENACT development we have been able to reduce the effort to prepare the Personal Health Gateway y exploring the ThingML, DivEnact and GeneSIS. Still, it is a significant issue be able to scale and efficiently manage large scale deployments of PHGs residing in patients homes. Therefore, we applied for further funding to address this complex tasks and got a new 4 year R&D project funded by the Norwegian Research Council to continue to investigate better solutions. The baseline is DivEnact and SINTEF (which has developed DivEnact) is partner in this new project. If we succeed with our ambitions in the new project (callend FLEET) we expect to reduce the effort by 500%
Improve solution quality and trustworthiness via more frequent Risk Analysis	Short time: Time for performing risk assessment reduced 10%	The reduction in terms of time to do risk assessment is not very significant, however the support for better coverage through automatic detection of vulnerabilities, risks and mitigation and the improved support for following up the risks through the integration with Jira is seen as even more significant

	Long time: Frequency of analysis increased from 2 to 6 per year	With better tool support the risk assessment will be conducted more often. This is also more and more a requirement from customers, as the risk picture vary dynamically and thus, continuous risk management are seen as more and more important
	Estimated Save cost of risk assessment by 10%	Cost saving is not significant; however, the improved risk management and the more frequent risk assessments are seen as even more significant
Provide the ability to process data on the gateway (CAAC)	Ability to store data to improve resilience of the service against connection lost. no data lost (not replaceable by later measurement, lose the important trend - accuracy), availability: Higher SLA: 95% - 97%	The CAAC enabler is facilitating the ability to store data om the edge and IoT devices as it supports the authorisation and context aware access control on these devices in terms of users and services. This again leads to a more robust system as the edge are able to handle data and events without being dependent on online connection to the cloud based back end.
	Ability to authenticate gateway in its context (0-1)	This ability is new and has not been supported before.

Table 6: Digital Health - Business KPIs

2.2.4 Conclusions

A set of ENACT enablers were integrated and tested on the Digital Health use case including KPI validation of the different enablers and execution of different validation scenarios, demonstrated the strengths and benefit of the ENACT tools for the use case. The main conclusions for the Digital health use case are listed below:

- The Orchestration and Continuous and Deployment enabler ThingML is successfully applied in the Development of the IoT and edge part of the Digital Health use case. The quality of the tool is mature enough as for Tellu to exploit this tool in its production chain.
- The Orchestration and Continuous and Deployment enabler Bundle with ThingML DivEnact and GeneSIS is targeting a significant problem that need to be encountered for a large scale deployment of the Digital Health use case. Currently the Use case is already commercialised and is deployed and applied for about 200 patients. However, to scale this to thousands of patients the efficiency of the onboarding, deployment, evolution and operation management need to be handled in and efficient way. The Orchestration and Continuous and Deployment enabler Bundle this issue and the current validation is promising. Therefore, we have successfully applied for a new R&D project funded by the Norwegian Research Council to further explore and develop this bundle to reach the goal of scalable, efficient and robust fleet management of large scale deployments of the Digital Health system
- The CAAC enabler enable to do authorisation and access control and to dynamically adjust according to the context. This is an innovative tool that fits with a set of scenarios that is very relevant for the Digital health use case, as authorisation and access control of devices and context aware access control to allow user access to edge devices according to the actual context is appearant as the Digital Health is concerned about monitoring the context situation and provide status information and alarms. Risk levels and escalation status is very dynamic and it is natural to adapt authorisation and access control based on context changes (e.g., fall detection) if that can be properly and efficiently controlled.
- The Risk Management enabler supports proper risk assessments based on the system architecture model and its dataflow model and it supports a build in expert system with questionnaires to automatically detect potential vulnerabilities, risks and mitigation and it integrates with issue management systems like Jira. We see this as a very interesting tool as it

will mature. In particular since compliance to standards such as ISO 27001 as well as regulations and laws for privacy and security in the eHealth domain is very strict.

- The Test and Simulation enabler provide support for simulation of edge and IoT devices as well as cloud based nodes, which represents a significant contribution in terms of more efficient and less costly testing of software systems that span across the IoT, edge and cloud infrastructures.

2.3 Smart Building

This section presents the validation and verification of the ENACT enablers in the Smart Building use case. During the first period of the project, several ENACT tools were successfully tested to enable continuous deployment, solving actuation conflicts and ensure security and privacy of the communications. During the second period, additional integration and adaptation has been carried out to identify risks at design time, correct behavioural drifts, enforce security and privacy and self-optimizing controller.

The Evaluation and Validation of ENACT results in the first period of the project applying the Smart Building use case included the evaluation and validation of the use case itself as a reference for the ENACT tools. The tested DevOps scenarios focused on the concurrent operation of multiple energy efficiency and user comfort IoT applications with orchestration of components, actuation conflicts, trustworthiness, security and privacy risks. In those DevOps scenarios, a set of the first versions of ENACT tools was tested: the Orchestration and Continuous Deployment enabler (GeneSIS), the Actuation Conflict Manager (ACM) enabler and the Security and Privacy Monitoring enabler.

In the second period, implementation of the DevOps scenarios was deepened to cover more cases. In this phase, the Risk-Management enabler, the Context Monitoring and Actuation Conflict Management enabler, the Online Learning enabler, and the Security & Privacy Control enabler were tested on the Smart Building case.

A summary of the evaluation and validation performed applying the Smart Building use case is shown in Table 7, including both first and second period results. The summary is provided in the context of the identified KPIs related to the Smart Building use case (as stated in D1.1).

Code	KPI	Enabler involved	Status
TO1.1	**Interface with Risk Management enabler.** The Orchestration enabler interacts with the Risk Management enabler to provide it with the list of devices selected as part of the SIS.	Risk Management enabler	Completed indirectly. The risk management tool provides the list of mitigation actions which asset (device in particular) needs to fulfil, therefore eliminating technically some of the devices at hand.
TO4.1	**Integration with SOFIA/SMOOL.** **Demonstrate the integration of the Orchestration and deployment enabler with the SMOOL platform:** (I) Demonstrate the continuous deployment of SMOOL client and their automatic integration with SMOOL broker. (II) Demonstrate data exchange of the deployed components via SMOOL.	Orchestration and Continuous and Deployment enabler	Completed. The SMOOL middleware was integrated via the Orchestration and deployment enabler (GeneSIS and ThingML) for continuous deployment and data exchange.

TO1.2	**Deployment of the use case applications** Demonstrate the continuous deployment of the two smart building applications. Including actuation conflict managers and S&P monitoring probes.	Orchestration and Continuous and Deployment enabler	Completed. Energy efficiency and user comfort IoT applications were continuously deployed and updated including (i) actuation conflict managers, (ii) security and privacy monitoring and controls (via SMOOL), and (iii) devices with no direct access to the Internet.
TO2.1	**Direct conflicts when acting on shared actuators.** The Conflicts enabler identifies and avoid conflicts in commands send to shared actuators by one or two IoT Systems or applications.	Context Monitoring and Actuation Conflict Management enabler	Completed. The Actuation Conflict Manager (ACM) enabler was implemented for thermal comfort control when two different IoT applications sent different temperature setpoints to the same heating actuator.
TO2.2	**Indirect conflicts when acting on shared physical entities.** The Conflicts enabler identifies and avoid conflicts when acting on shared physical entities, e.g. lighting, by two or more IoT Systems even if actuators are not directly shared.	Context Monitoring and Actuation Conflict Management enabler	Completed. A demo was implemented with two different IoT applications that indirectly affected to lighting physical entity. The Behavioural Drift Analysis (BDA) enabler was used to evaluate how much the observed behaviours of the IoT System are different from the expected ones.

TO2.3	**Online learning and user comfort and energy consumption** Demonstrate a positive impact on the user comfort and energy consumption by comparing how the system evolve with or without the online learning	Online Learning enabler	Completed. The Online Learning Enable has been implemented for the KUBIK building and demonstrated improved comfort and lower energy consumption compared to a standard thermostat. Further details can be found in D3.3
TO2.4	**Online Learning and behavioral drift analysis**[3] Demonstrate integration between the online learning tool and the behavioral drift analysis enabler	Online Learning enabler	The integrate BDA and OLE were integrated so that the signals (state variables) of the OLE are used by the BDA to compute a behavioral drift signal, which is then fed back into the Online learning tool. Further details can be found in D3.3
TO3.1	**Risk level.** Risk enabler indicates a risk level in a particular configuration of the IoT system	Risk Management enabler	Completed. Different IoT system settings were tested to evaluate the different levels of risk.
TO3.2	**Risk level analysis combining legacy systems.** Risk enabler to analyse possible threats brought by new system parts and its combination with legacy systems.	Risk Management enabler	Completed. New sensors were added to KUBIK to test changes in the Risk-Management enabler
TO3.3	**Risk Management to minimize risks.** Risk-Management enabler support selection of the best IoT system elements that minimize risks according to the user requirements.	Risk Management enabler	Completed. The Risk-Management enabler was used to select the configuration of the new integrated sensors.
TO3.4	**Risk level change alert.** DSS enabler raises alerts whenever a change in the IoT system (e.g., a redeployment) modifies the risk level.	Risk Management enabler	Completed. Alert were sent when a change in the risk level was detected.

[3] The ACM was replaced by the BDA enabler compared to D1.3, as it is more coherent with the OLE application.

TO3.5	**Demonstrate configurability of alarm threshold.** Demonstrate that the S&P Monitoring enabler enables the user to set the desired thresholds to raise cybersecurity alarms.	Security & Privacy Monitoring enabler	Completed. The Security and Privacy Monitoring enabler was used to raise cybersecurity alarms when messages were sent by unauthorized nodes and abnormal network traffic was detected
TO3.6	**Security enforcement. Demonstrate reaction to attacks and incidents.** Demonstrate that the S&P Monitoring enabler work together with S&P Control enabler which helps reacting to attacks or incidents.	Security & Privacy Control enabler	Completed. The Security & Privacy Control enabler as extensions to SMOOL server and KPs was used to enforce in the when cybersecurity threats were detected.

Table 7: Smart Building - ENACT Tools KPIs

The KPIs described above are matched with the following tests.

2.3.1 Evaluation and Validation DevOps Scenarios

In addition to the evaluation and validation KPIs listed in the previous section, a set of DevOps scenarios related to the Smart Building use case are aggregated in sets of tests. These tests are intended to validate the various the aspects encountered by the DevOps scenarios. The DevOps scenarios and the related set of tests were described in D1.1. During the second period the focus has been on the testing of the GeneSIS enabler, Risk-Driven decision support enabler, the Behavioural Drift Analysis (BDA) enabler, the Online Learning enabler and the Security and Privacy Monitoring and Control enabler. Table 8 summarizes the status of the tests designed for the Smart Building use case.

Group of tests	Test	Description	ENACT tool	Test Update	Status
Thermal comfort control – Heating design tests	Test 3.1.0.1 Risk-driven heating design	Test risk during the design of thermal comfort application design	Risk Management enabler	The Risk Management tool was used to read analyse the proposals of the architecture.	Accomplished. A list of mitigation actions requiring specific technical capabilities was proposed on the tool.

	Test 3.1.0.2 ThingML models integration in the SMOOL middleware	Test to verify if ThingML models integrated in SMOOL are automatically deployed	ThingML and Orchestration and Continuous Deployment enabler	The Orchestration and Continuous Deployment enabler was added to the heating design tests.	Accomplished. SMOOL client code was integrated with ThingML models and was automatically and correctly deployed by GeneSIS.
Thermal comfort control – Conflict in heating actuator use test	Test 3.2.0.1 Actuation conflicts on the use of the same actuator	Test of the capability of the ACM tool to resolve the sending of conflicting orders by different IoT flows to the same heating actuator	Actuation Conflict Management (ACM) enabler	The Actuation Conflict enabler was tested for user comfort scenario, both in thermal control and lighting demo.	Accomplished. The ACM enabler rewrote the IoT flows for an adequate behaviour
	Test 3.2.0.2 Integration of the Actuation Conflict Manager enabler with GeneSIS	Test to verify if the ACM tool is correctly deployed	Orchestration and Continuous Deployment enabler	The Orchestration and Continuous Deployment enabler was added to the conflict in heating actuator use test	Accomplished. GeneSIS deployment model is properly consumed by ACM tool, which successfully use it to identify and solve actuation conflicts. The resulting actuation conflict managers are correctly deployed back with GeneSIS
User comfort control – Conflict in lightning actuation tests	Test 3.3.0.1 Actuation conflicts on the same physical variable	Test actuation conflict on the same control parameter	Behavioural Drift Analysis (BDA) enabler	The Behavioural Drift Analysis enabler was implemented on a brightness actuation scenario.	Accomplished. The Behavioural Drift Analysis (BDA) was successfully demonstrated to observe drifting behaviours (like luminosity ones in Kubik and sound in Smart Home).
	Test 3.3.0.2 Integration of the Behavioural Drift Analysis enabler with GeneSIS	Test to verify if the BDA tool was correctly deployed	Orchestration and Continuous Deployment enabler	The Orchestration and Continuous Deployment enabler was added to the conflict in lightning actuation tests	Accomplished. The BDA tool and its required monitoring probes are correctly deployed with GeneSIS.

Smart building alerts for user comfort tests	Test 3.4.0.1 Trustworthy smart building alerts	Test of the capability of the Security and Privacy Monitoring tool to check if the messages were maliciously modified or sent by unauthorized nodes	Security and Privacy Monitoring enabler	Alarms from SMOOL security KP added to Security and Privacy Monitoring enabler dashboard.	Accomplished. The Security and Privacy Monitoring enabler detected abnormal network traffic and was able to show alarms when issues arise in the security metadata of the communications.
	Test 3.4.0.2 Improved security in smart building alerts	Test to implement proactive security measures	Security and Privacy Control enabler	Security and Privacy Control enabler (through SMOOL) added to smart building security test	Accomplished. Communication was successfully cut when threats were identified.
Thermal comfort control – Self-optimizing controller design	Test 3.5.0.1 Self-optimizing controller	Self-optimization test of HVAC operation	Online Learning enabler	The Online Learning enabler was successfully applied to the Self-optimising controller case	Accomplished. The OLE showed improvement of the controller
	Test 3.5.0.2 Integration of the Online Learning enabler with GeneSIS	Test to verify if the Online Learning tool was correctly deployed	Orchestration and Continuous Deployment enabler		Accomplished. The OLE (and the simulator) have been successfully deployed with GeneSIS.

Table 8: Smart Building - ENACT Tools Validation and Verification Test Progress

The tests were successfully completed during the second period of the project.

2.3.2 Evaluation and Validation Application Scenarios

The Smart Building use case comprises two complementary scenarios with the evaluation of different enablers in each scenario.

2.3.2.1 The Smart Home scenario

The Smart Home system developed and operated in this scenario leverages an infrastructure mixing facilities from Kubik and the Smart Home laboratory developed by CNRS. The system aims at improving user comfort and daily routines in the building and is formed by three different sub-systems as illustrated by the informal diagram shown in Figure 68.

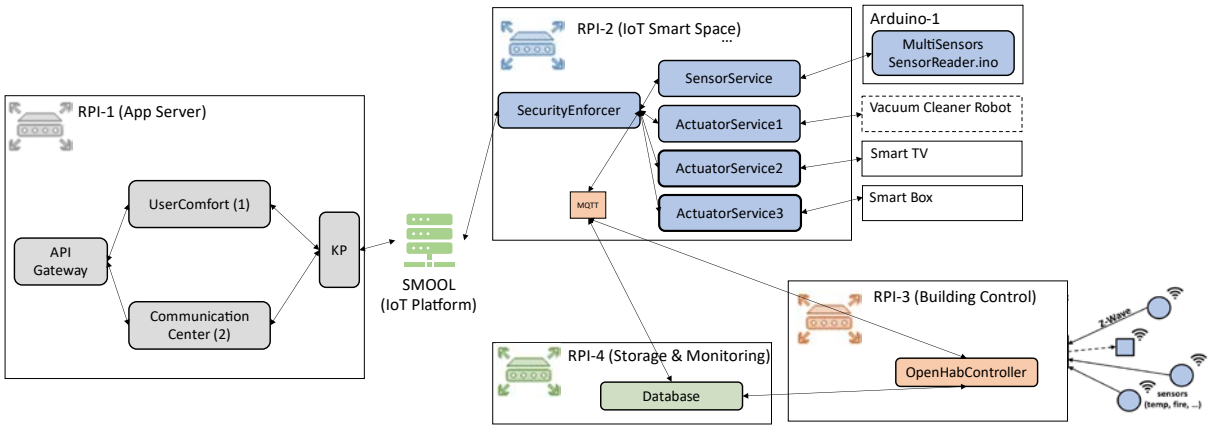

Figure 68: Architecture of the Smart Home

The first system (namely IoT Smart Space in) includes a gateway that connects to *(i)* smart devices (*i.e.*, Arduinos, Smart TV, Smart TV Box), and *(iii)* other Raspberry Pis connected to microphone and video camera, respectively, for audio and video processing. Whilst the Arduinos are physically connected to the gateway using serial port, the Smart TV, Smart TV Box, and Raspberry Pis are connected via WiFi and can only be accessed from the Local Area Network. The second system (namely Building Control) connects and controls a wireless sensor/actuator network using the Z-Wave protocol. These two systems form the internal of the smart home systems and provide applications with access to sensors and actuators. They connect to a third gateway (a Raspberry Pi) used to stored and monitor the overall system.

Finally, the third system (namely App-Server) hosts applications that receives sensor data from the sensors and sends actuation commands to the actuators available in the other two systems. The App-Server interoperates with the IoT Smart Space and the building control using a Cloud-based instance of the Interoperability SMOOL IoT middleware[4] that has a semantic broker for connecting heterogeneous devices or sources of information. More precisely, the App-Server only communicates with the IoT Smart Space, which in turn forwards authorized and secured messages to the Building Control system. For security reason, all the messages between the App-Server and the IoT Smart Space are checked and controlled by the SecurityEnforcer component deployed on RPI-2. All the applications deployed on the App-Server are accessible through an API gateway ensuring their secured remote access. The two applications developed and operated in this scenario using the ENACT enablers are:

- The user comfort application aims at (i) controlling luminosity level and maximizing the exploitation of daylights; (ii) regulates the in-door temperature; (iii) enables basic management of multimedia devices; (iv) manage user daily routines.
- The Communication Control applications aims at controlling the audio devices within the home and ensure an acceptable ambient acoustic level in different situations (e.g., as little disturbing sounds during phone calls).

Figure 69 provides an overview of the infrastructure used in the scenario. To summarize, the scenario involves 51 devices (176 sensors and 69 actuators), 6 services and 6 edge nodes, 6 different communication protocols exchanging data through 576 channels, and 81 software components need to be deployed.

[4] https://bitbucket.org/jasonjxm/smool

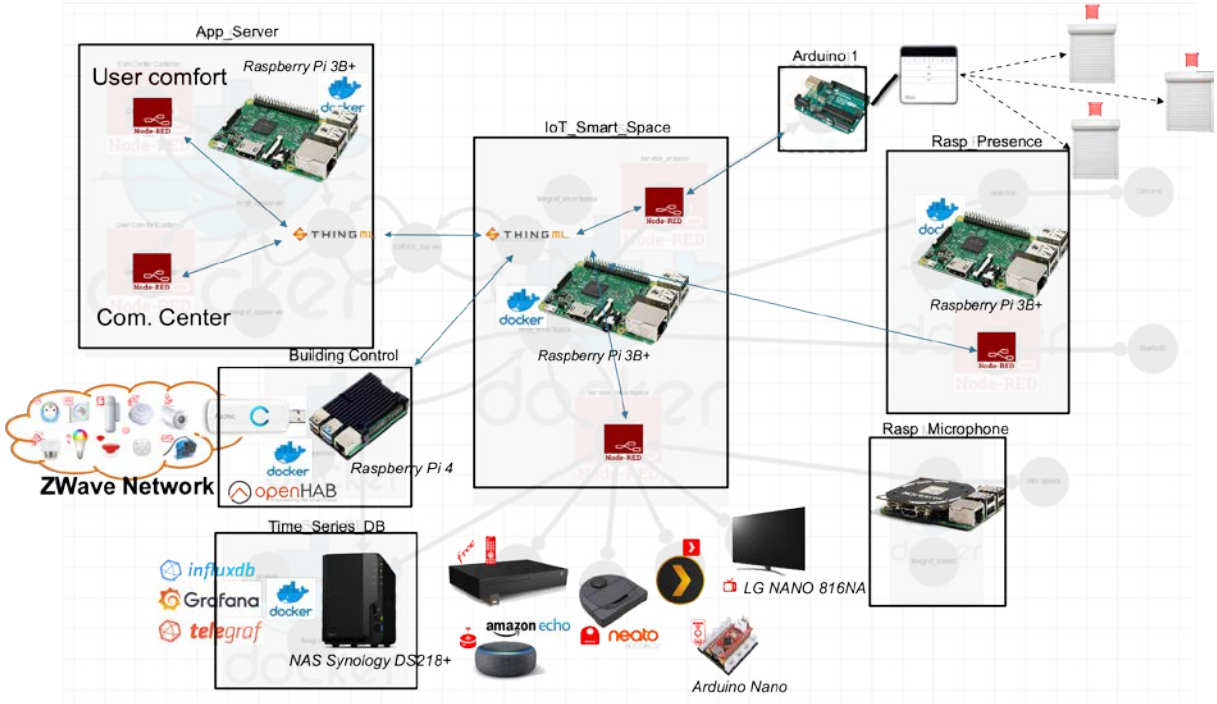

Figure 69. Infrastructure for the Smart Home scenario

The scenario defined to develop and operate the Smart Home system presented above involves 3 DevOps iterations (i.e., the delivery of three versions of the system, each delivery improving over the last before). In the following we detail these three iterations and how the ENACT enablers were used for the development and operation of the system.

2.3.2.1.1 First DevOps iteration

We started the scenario considering a first version of the user comfort application was delivered, tested and ready for deployment as well as all the software components for the IoT_Smart_Space and Building Control subsystems. The application is written using Node-Red and interact with SMOOL to gather sensors data and trigger actuation via a SMOOL client written using ThingML. The IoT_Smart_Space and Building Control sub-systems also interact with SMOOL using SMOOL client written in ThingML. For short, these systems include: software components to be deployed as docker containers, software components written in ThingML, Arduino sketchs, and software components to be deployed using SSH.

To deploy the application and the two sub-systems we used **GeneSIS**. The corresponding deployment model is shown in Figure 70 and can be found at the following address: https://gitlab.com/enact/smarthome-demo/-/blob/master/genesis/deployment-model-template-step1.json.

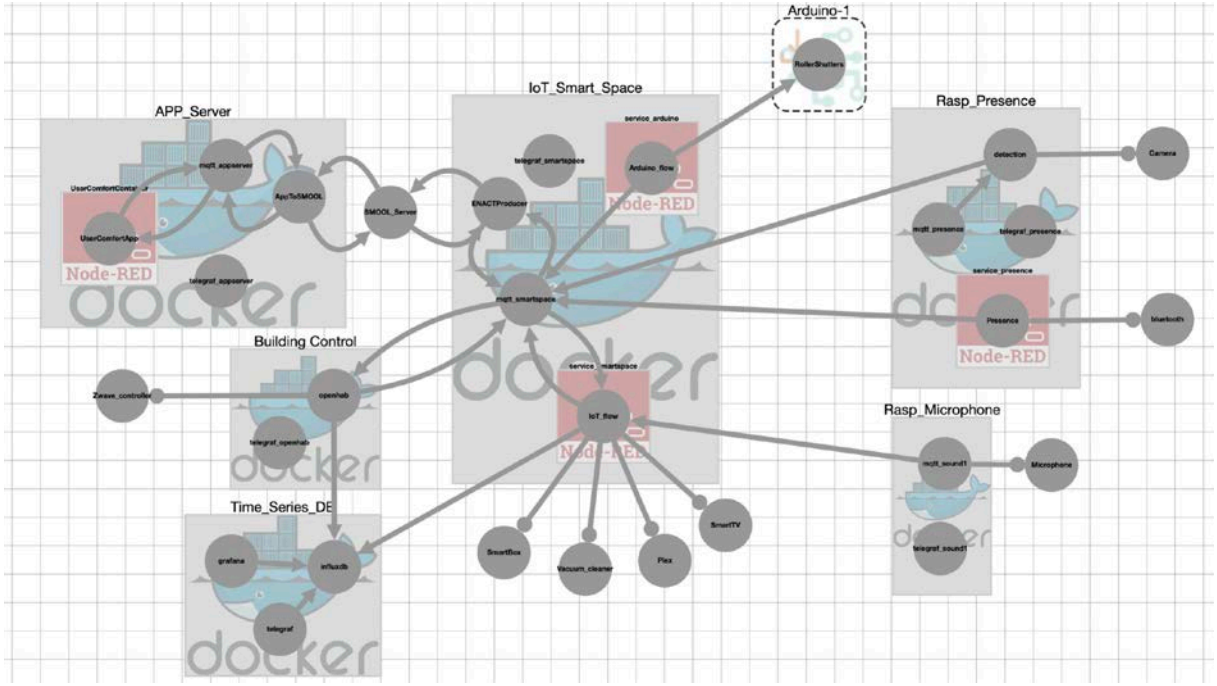

Figure 70. Initial deployment of the Smart Home system

By successfully validating this deployment we demonstrated the:

- The use of GeneSIS as a single tool to specify deployment over IoT, Edge and Cloud infrastructure.
- Deployment on devices with no direct access to Internet or only accessible locally (Arduinos, Raspberry Pi controlling the Microphone and the Camera).
- The ability to inject security policies in SMOOL by design. Indeed, within the deployment model, GeneSIS configure the default SMOOL security checker to ensure only authorized actuation orders from the SMOOL client can be send.

Once the application is deployed, the **Behavioural Drift Analyser** is used in order to monitor the behaviour of the system on particular control the proper management of the light in the smart home. The corresponding behavioural model is shown in Figure 71 can be found at the following address: https://gitlab.com/enact/smarthome-demo/-/tree/master/bda/luminosity

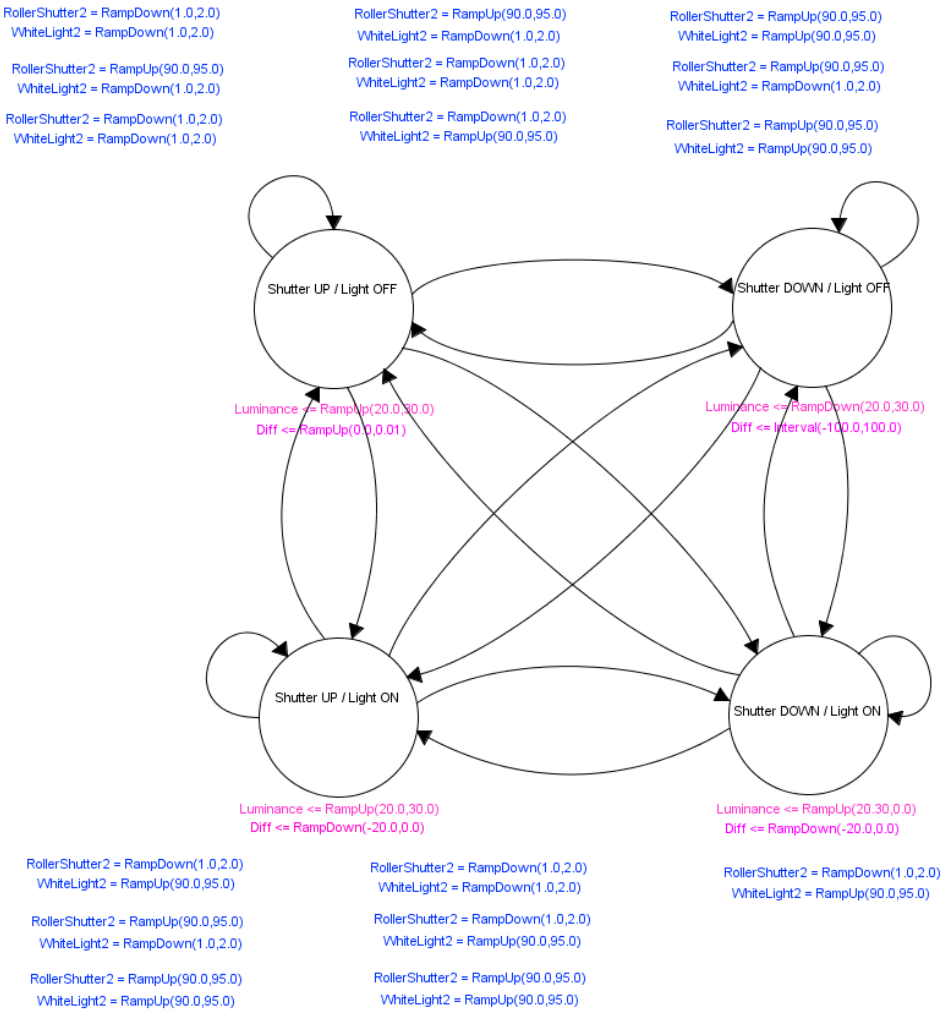

Figure 71. Behavioural model for the light management

By successfully monitoring the behaviour of the system we demonstrated the:

- Ability to monitor the effect of the application on the environment and to compute a behavioural drift (i.e., the extent to which the system behaves as expected)
- Ability to design a behavioural model involving multiple actuators and sensors.

At this stage no drift was observed.

2.3.2.1.2 Second DevOps Loop

During this iteration, a second application (i.e., CommunicationCenterApps) was designed, creating a home-working environment where the focus is put on controlling sound sources so as to prevent home workers to be disturbed during phone calls or video conferences (e.g., App_Phone_TV mutes TV while a phone call is in progress). The application is also implemented using Node-RED. Together with the new application it is also decided to use a more advance and fine-grained access control solution (jCasbin) to be deployed locally in the smart home.

Before the Communication Center Application is considered as delivered, it is tested against data from the real system. The Test and Simulation enabler is thus used to record sensor and actuator data from the system already deployed, thus preparing the datasets that will be used to test the new application. Test campaigns are prepared (see Figure 72) and rules added so to generate erroneous and malicious data. In particular, generating values out of range and values out of regular range. Configuration of the test campaign can be found here: https://gitlab.com/enact/smarthome-demo/-/blob/master/TaS/data/models/SmartHome%20Replayer%20Remotely.json

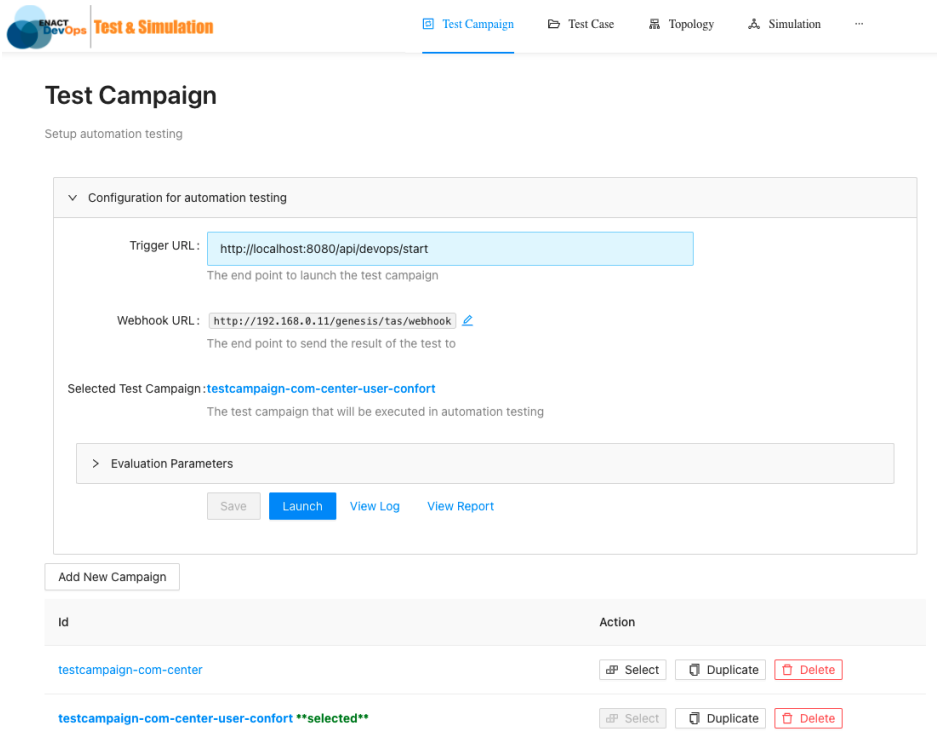

Figure 72. Trigger the Communication Center test campaign

By successfully testing the application we demonstrated the ability of the TaS enabler:

- To record and replay data from the real system, thereby simulating it.
- To generate and inject erroneous and malicious data during the tests.
- To build, trigger and present the results of test campaigns.

The test being passed successfully, the first deployment model is thus updated as depicted in Figure 73 to include these two new software components (see blue rectangles in Figure 73) and can be found at the following address: https://gitlab.com/enact/smarthome-demo/-/blob/master/genesis/deployment-model-template-step2-casbin.json

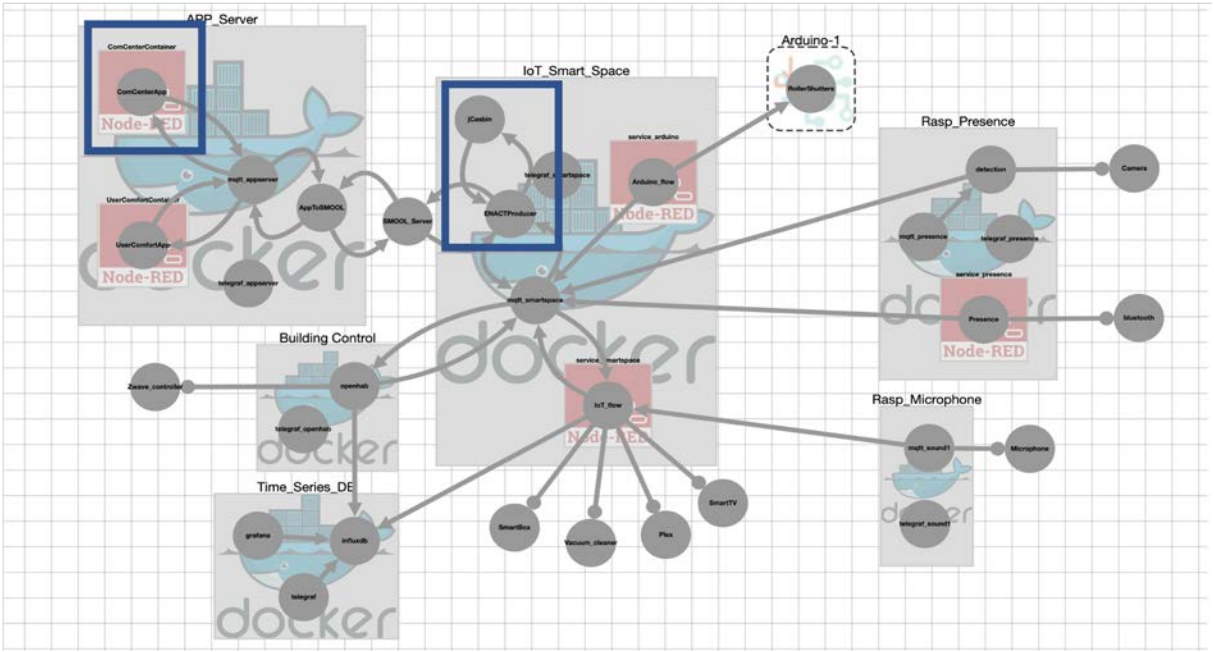

Figure 73. Second deployment model including the Communication Control application and jCasbin

As part of the DevOps pipeline implemented for this scenario, the system is checked against actuation conflict before deployment. The **Actuation Conflict Manager** was thus used to identify and solve actuation conflicts, relying on the WIMAC model depicted in Figure 74 and built from the GeneSIS deployment model together with the application and physical environment models. At this point, a potential direct actuation conflict was identified related to the concurrent access to the TV by the two applications (UserComfortApp for remote control of the TV and the CommunicationCenterApp to stop the TV when a phone call occurs). The management of this direct actuation conflict was straightforward as the applications merely send a Boolean value to a shared actuator. This conflict was solved by selecting among the available off-the-shelf actuation conflicts managers, in our case a component implementing a OR logic.

By successfully solving actuation conflict we demonstrated the:

- The ability of the Actuation Conflict Manager to identify direct actuation conflicts, also demonstrating that WIMAC models contain all necessary information for it.
- The ability of the Actuation Conflict Manager to provide developers with off-the-shelf actuation conflict managers to solve actuation conflicts
- The ability of the Actuation Conflict Manager to inject actuation conflict managers into a Smart IoT System.
- Complementarity of the GeneSIS and ACM enablers, GeneSIS provides the basis for the Actuation Conflict Management enabler to build a WIMAC model.

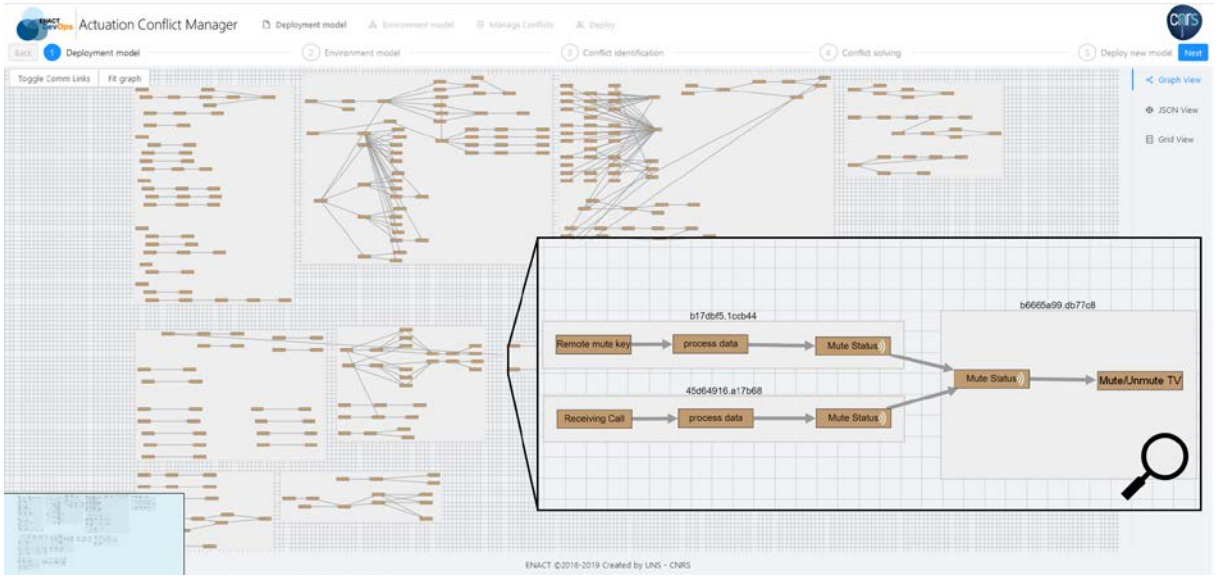

Figure 74. WIMAC model of the Smart Home system

Once this conflict solved and the application automatically adapted by the Actuation Conflict Management enabler, the deployment was performed using GeneSIS.

By successfully deploying the new version of the system we demonstrated the:

- Ability of GeneSIS to adapt a deployment. Only redeploying the software components updated (in this case the components in the blue rectangle in Figure 73).
- Ability of GeneSIS to deploy a third-party security mechanism (jCasbin, available as an off-the-shelf GeneSIS component) and to update the SMOOL security policy to seamlessly integrate it.
- Complementarity of the GeneSIS and ACM enablers, GeneSIS provides ACM with the ability to deploy the actuation conflict managers.
- Complementarity of the GeneSIS and TaS enablers, tests campaigns were automatically triggered once the deployment is completed to ensure non-regression testing.

2.3.2.1.3 Third DevOps Loop

In the third stage, a new vacuum cleaner robot is added into the IoT Smart Space. The UserComfort application is modified with the objective to trigger cleanings at scheduled times. We do not provide details here but, similar to the former step, the application was deployed using GeneSIS and tested with the TaS enabler.

In addition, BDA enabler is configured with the objective to assess the effectiveness of the SIS in managing the TV. In particular, looking at the operating status of the TV and at the communication status (i.e., OFF/ON means TV:ON,COM:OFF). The configuration ON/ON is not legitimate as the direct ACM instantiated is supposed to mute the TV while a communication is in progress. The model used is depicted in Figure 75 and available at: https://gitlab.com/enact/smarthome-demo/-/tree/master/bda/noise

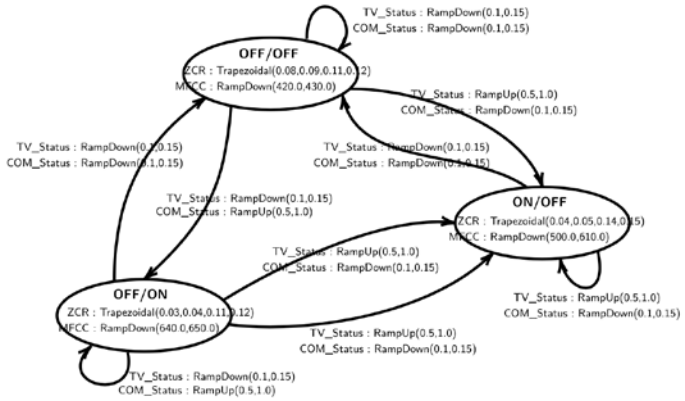

Figure 75. Model of the expected behaviour of the SIS when managing the TV and the communications

The BDA enabler allowed us to observe the following unexpected behaviour as depicted in Figure 76. While the SIS was under operation, an autonomous smart vacuum cleaner moving around in the house, bursts into the living-room. By operating in the living-room, this device produces unexpected sounds resulting in behavioural drifts. The reported behavioural drift suggested the presence of an indirect conflict on the sound physical property (see Figure 77) that could not be identified by the ACM because its model of the smart home environment was not updated after the introduction of the vacuum cleaner.

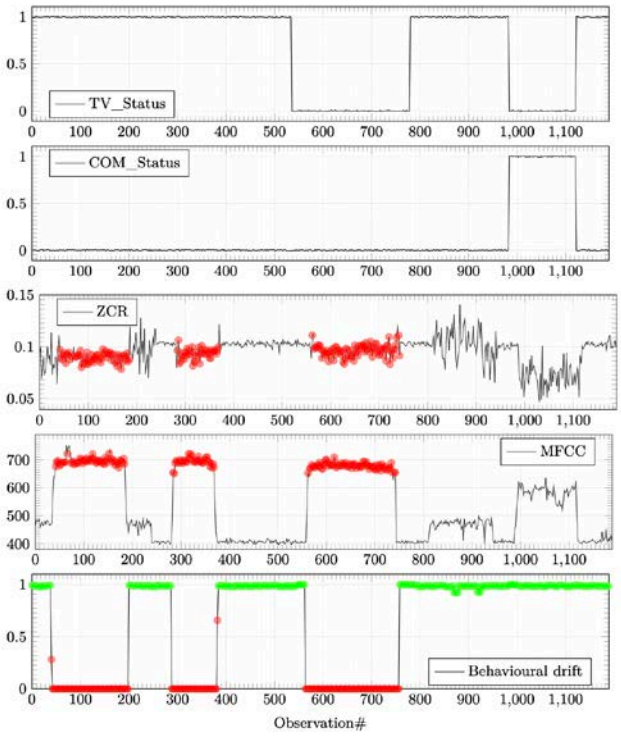

Figure 76. Behavioural drift in the management of the acoustic level

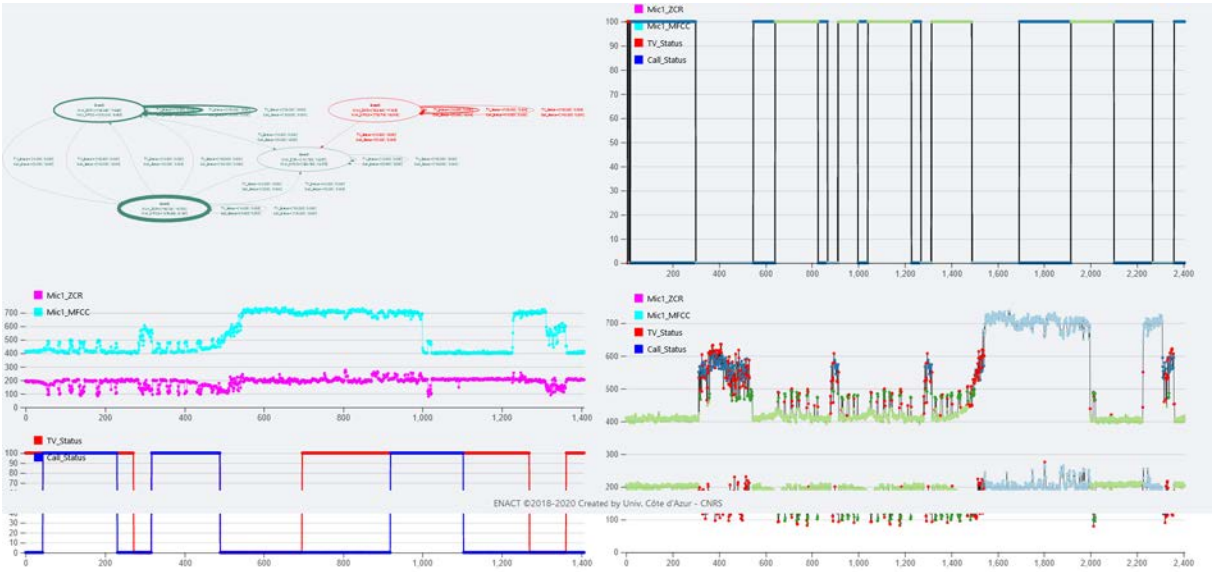

Figure 77: Behavioural Analysis identifying the drift symptoms

By successfully identifying and analysing a behavioural drift we demonstrated the:

- The ability of BDA to monitor multiple behaviours (sound and light management behaviours)
- The ability of BDA to identify drift in the behaviour of a SIS during operation.
- The ability of BDA to help identifying the symptoms of a drift in the behaviour of the SIS.

A new development cycle was therefore necessary to correct the model of the physical environment. A new WIMAC model was produced from the deployment, application, and environment models of the SIS under operation (benefiting from the ability of GeneSIS to maintain up-to-date a model of the SIS deployed). The environment model was updated, specifying the impact of the vacuum cleaner on the audio of the living room and an indirect actuation conflict could be detected. A specific Actuation Conflict Manager was designed to solve this conflict (see Figure 78) and the new SIS was deployed. The code of the ACM can be found at the following address: https://gitlab.com/enact/smarthome-demo/-/blob/master/acm/acm_eca.eca

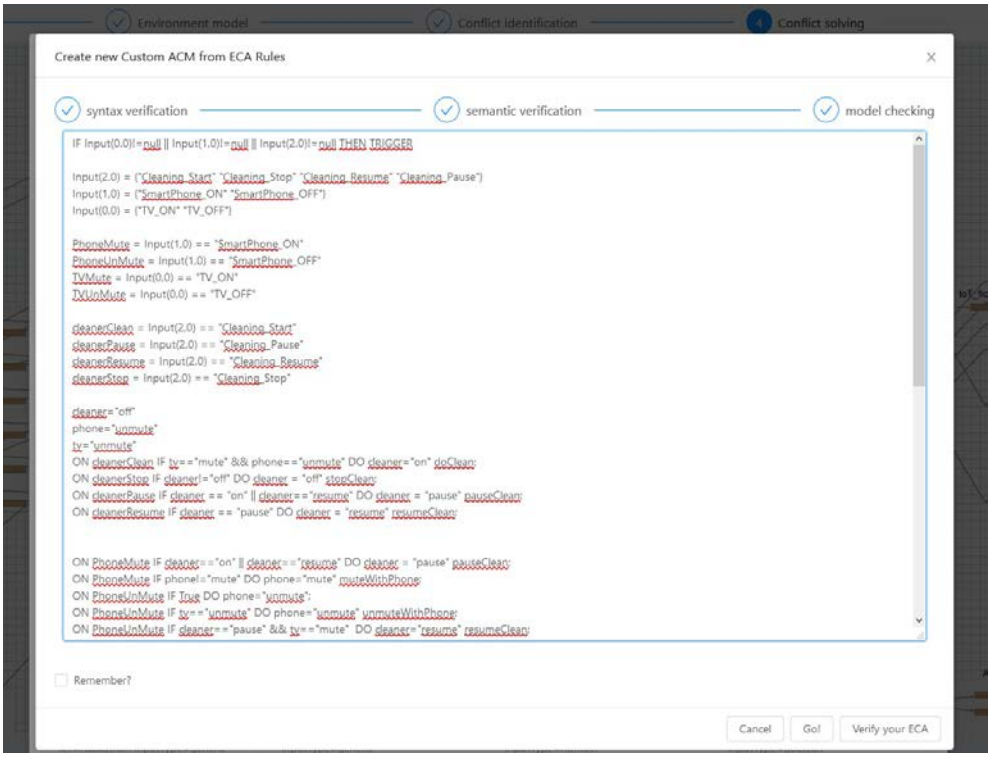

Figure 78. Custom Actuation Conflict Manager

By successfully solving this conflict we demonstrated the:

- The ability of the ACM enabler to identify indirect actuation conflicts.
- The ability to specify custom actuation conflict managers using the ECA + language, as well as the ability to check custom ACM against temporal and logical properties before generating the corresponding ACM implementation.

2.3.2.2 The Kubik facility scenario

The Energy Efficient Building application was implemented as a benchmark for ENACT tools and was 100% operational in the first period of the project. The Energy Efficient Building application performs thermal comfort control of the building using environmental sensors, motorized blinds, electric heaters and HVAC actuators. This application allowed to test the following tools developed in the ENACT project: Orchestration and Continuous Deployment enabler (GeneSIS), Actuation Conflict Manager (ACM) enabler, Behavioural Drift Analysis (BDA) enabler, Online Learning enabler, Risk Management enabler, Security and Privacy Monitoring enabler, Security and Privacy Control enabler.

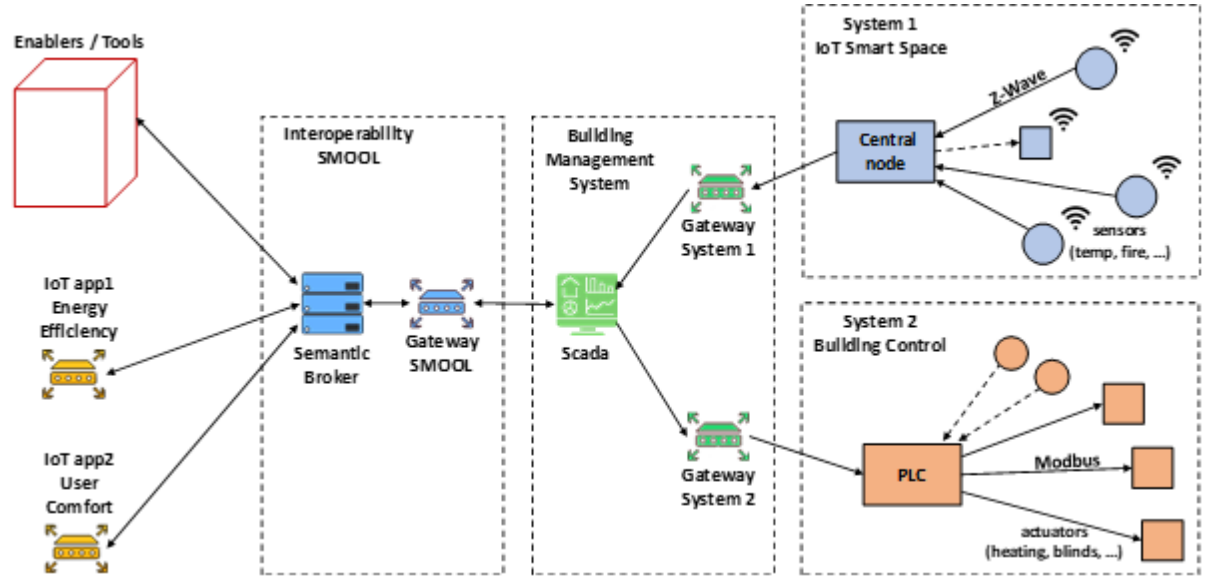

Figure 79: Smart Building – Schema of the Energy Efficient Building application

To be able to implement the different enablers inside the project, an integration with the SMOOL architecture has been carried out in order to obtain data from different sensors (Z-Wave or PLC). This architecture design is based on the physical components installed in KUBIK building. In previous deliverables, an explanation of the technical architectures around Z-Wave and PLC was described. In this deliverable, the software integration between SMOOL and KUBIK building is going to be described in order to explain how the secure actuation through SMOOL is implemented.

Figure 80 depicts the architecture for secure actuation implemented in the KUBIK building. As it can be seen, the "Deployed Sensors" part represents the acquisition process in which a set of sensors using both Z-Wave (system 1) and legacy PLC (system 2) obtains data measures to be sent to a middle device called "Mosquitto MQTT broker"[5].

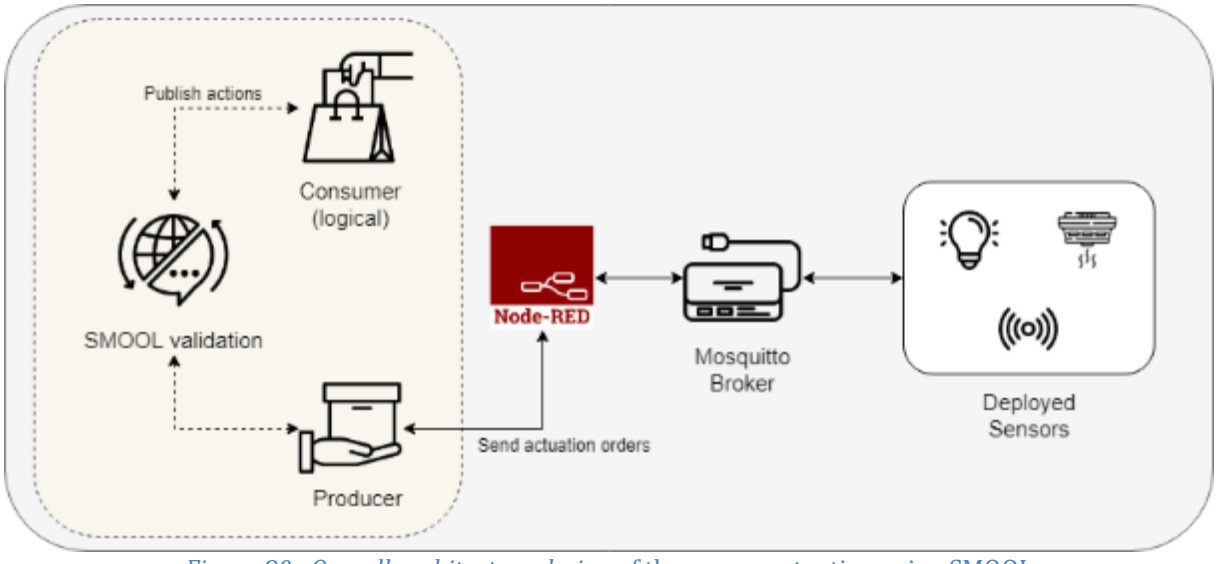

Figure 80: Overall architecture design of the secure actuation using SMOOL

The Mosquitto broker uses MQTT messages by means of which a program can subscribe to different sensors to obtain values in real time. In the proposed case, the software used to make a subscription is Node-RED software which reads the data from the MQTT broker and publish it to SMOOL. Basically Node-RED code accomplishes two tasks: 1) it can read data from Mosquitto MQTT broker by making

[5] https://mosquitto.org/

a direct subscription of different deployed sensors and publish it to SMOOL; 2) it can play as the backbone to allow to SMOOL middleware send direct actuations.

The Producer of SMOOL middleware is used to send secure actuations when it is required by using a secure token to the Consumer. The aim of the Consumer module in secure actuation is to execute a set of different expert rules using the obtained data from the Producer. For example, if we try to send a secure command to open the blinds at night, expert rules may deny this actuation because for security reasons it makes no sense to raise the blinds when it is dark outside and neither light nor heat can enter the room.

As explained, the data acquisition and actuations processes are carried out by different software modules. The main components which make the interoperability between the different modules is, Z-Wave gateway, PLC gateway and SMOOL middleware. The translation between MQTT messages from both gateways to SMOOL messages is done using Node-RED code.

At low level in the presented Smart Building Architecture, the data acquisition and actuation processes are managed by a set of hub systems (called Gateways in the architecture). These hubs are configured to allow the direct communication between different sensors/actuations and a MQTT broker called Mosquitto. Each hub uses its own communication protocol to acquire or send actuations to the target device. For example, one hub is configured to manage only Z-Wave communications towards connected devices and another hub is configured with MODBUS communication protocol.

These hub systems use an internal operating system called JEEDOM which makes possible a graphical configuration of the connected devices and external services. In this regard, each hub is configured to make a direct connection to Mosquitto MQTT broker. Figure 81 shows an example of how JEEDOM is configured to send the data reader from a set of Z-Wave devices directly to the Mosquitto MQTT broker.

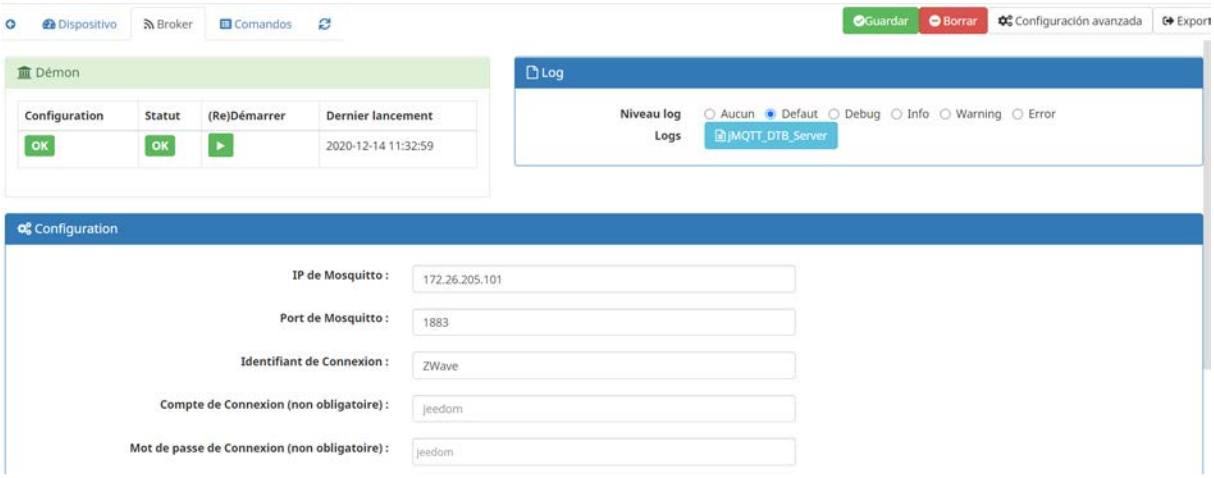

Figure 81: JEEDOM hub configuration to connect Mosquitto

The Mosquitto MQTT broker creates a subscription channel for each connected device coming from the hub. This channel will be used as a "gate" to allow to external programs, read online data coming from sensors, or set variables to perform actuations. More information about how is working Eclipse Mosquitto software can be found in their official page[6].

Once the connections between the Z-Wave/PLC hub (JEEDOM) and Mosquitto broker are established, the next step is to configure Node-RED programming tool. This tool provides an online editor (formed by boxes and connections) to configure action flows. Each action flow allows the intercommunication between external programs and deployed sensors in an easy way by making a combination of REST API based endpoints and MQTT broker subscriptions. Figure 82 depicts the configuration parameters

[6] https://mosquitto.org/

needed to connect Node-RED with Mosquitto broker and allow the configuration of different workflows to make the required subscriptions to each connected device.

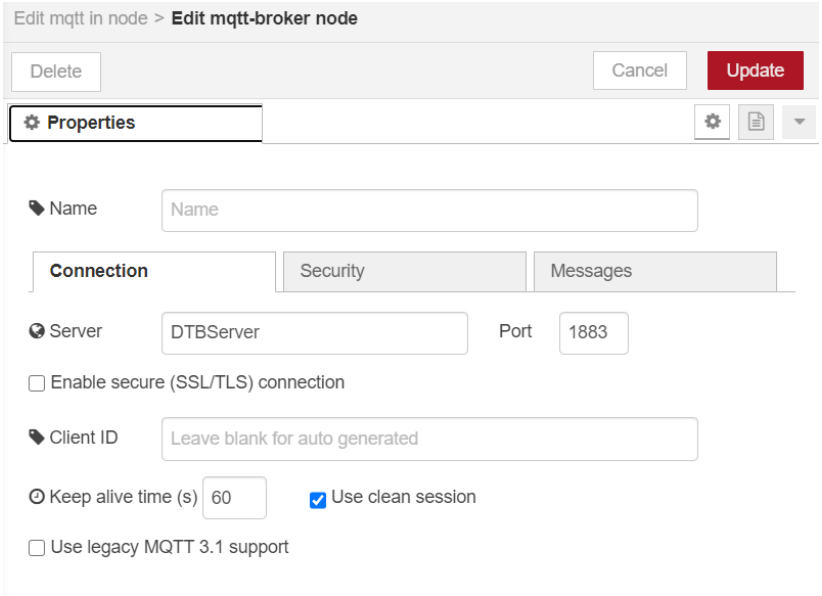

Figure 82: NODE-RED configuration to Mosquitto broker

2.3.2.2.1 DevOps scenario for Security and Privacy Control enabler

The following scenario of secure actuation enforcement was implemented to test the SMOOL capability to intervene the communications of the smart things in the smart building when issues arise in the security metadata of the messages.

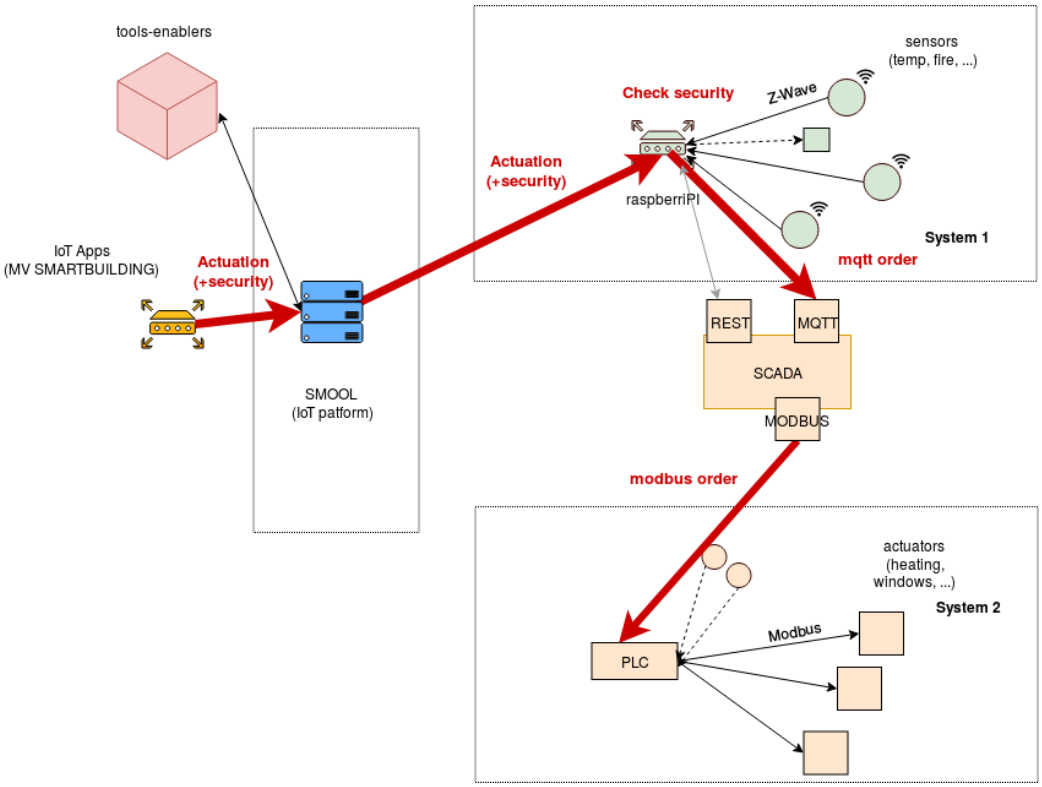

Figure 83: SMOOL IoT Platform control on actuation messages

In the figure, the Consumer KP reads data from the sensors in the Kubik and generates the actuation orders adding the required security token, and sends them to SMOOL server. The Producer KP gets the actuation orders from SMOOL server, and sends them to the corresponding actuators.

After having the logical connection between Node-RED and Mosquitto, it is necessary to model the integration with SMOOL KPs. To do so, it is necessary to configure the required Node-RED action flows to allow the data acquisition process where the MQTT broker information gets to the SMOOL Producer KP. This enables the Producer KP to build messages where the payload value is one of the MQTT broker topics, i.e. sensor values.

Figure 84 depicts the acquisition process flow configured in Node-RED to read the data from Mosquitto broker and send it through a direct socket connection to the SMOOL Producer KP. Each connected line represents a sensor inside the KUBIK building. For clarity convenience, only a portion of the whole model is showed in the figure, because multitude of sensors are deployed in the KUBIK building and each one has its own action flow.

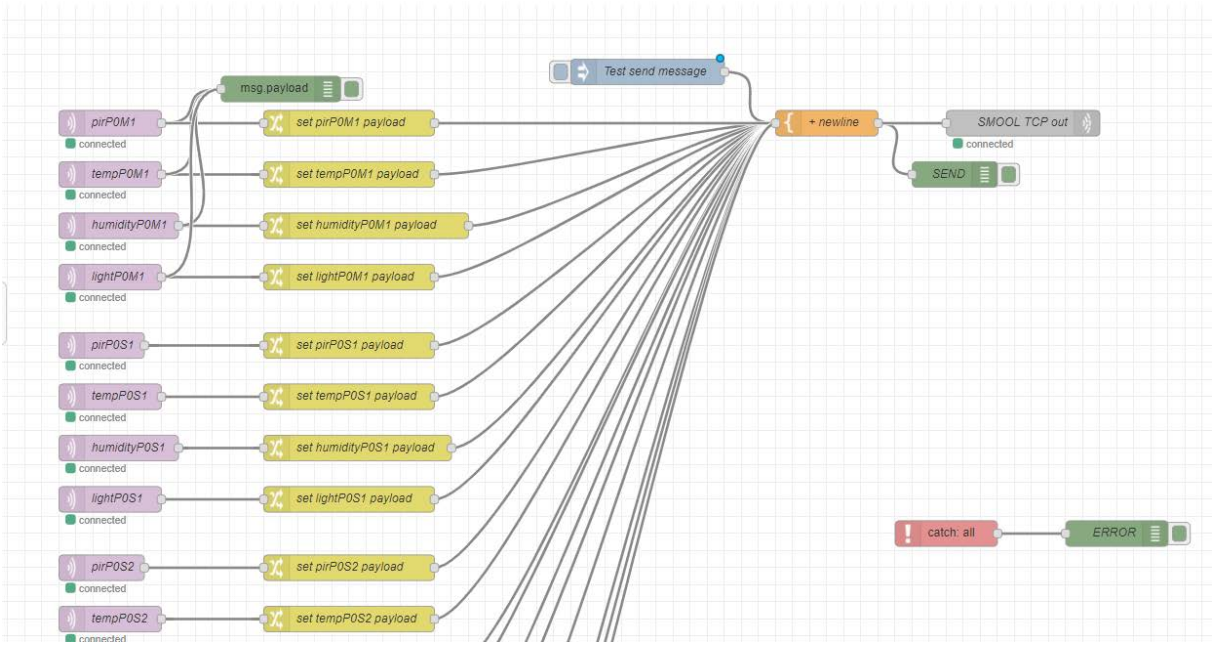

Figure 84: Node-RED flows from the MQTT client to SMOOL Producer KP

Figure 85 depicts the secure actuation simulation process in which it is possible to see two different action flows: 1) the flow configured to send from Node-Red to SMOOL Consumer KP the available orders so as the Consumer KP can generate the actuation orders with the security token (top of the image); 2) the flow to publish the actuation orders from Produce KP towards the actuator (low part of the image).

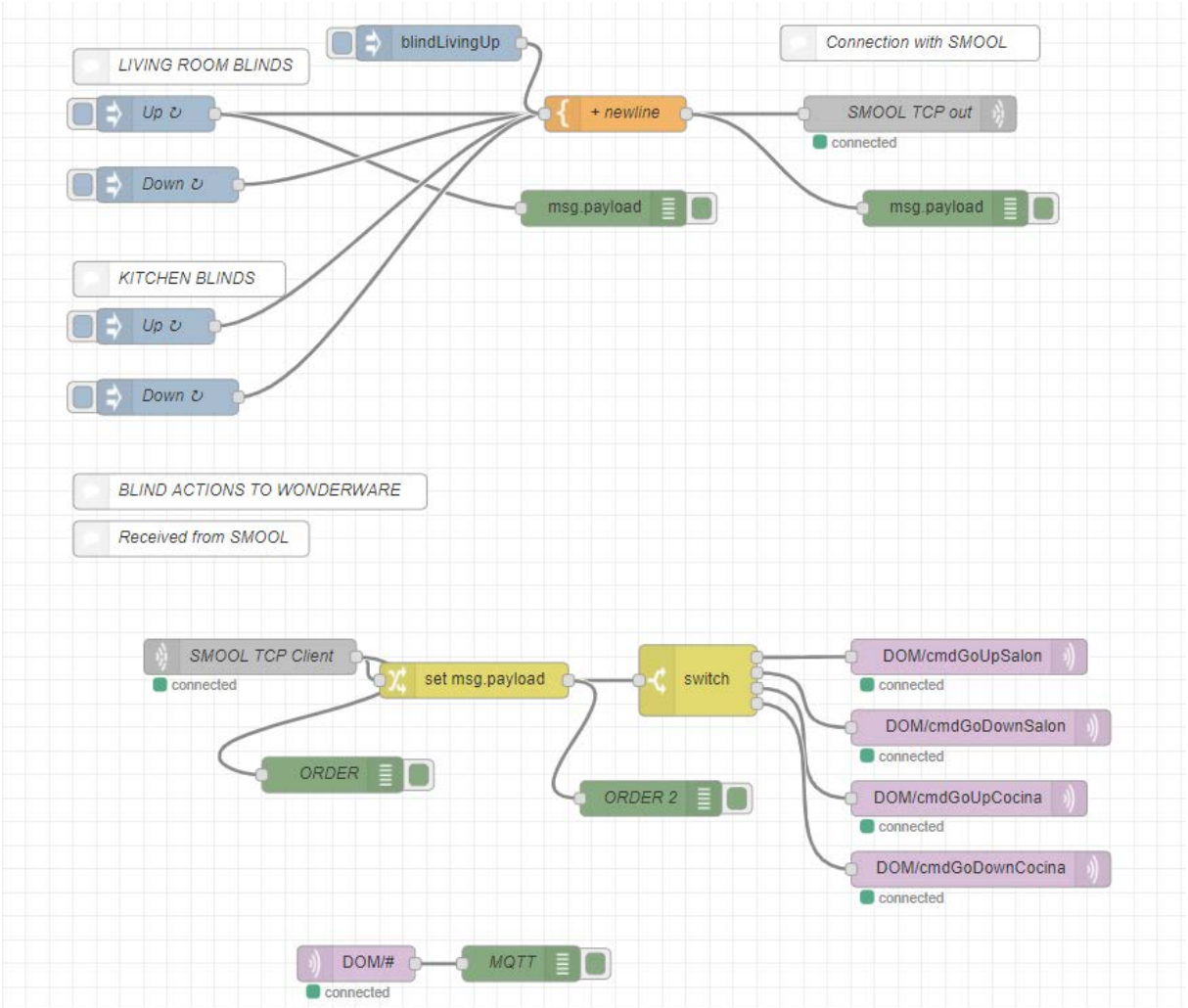

Figure 85: Node-Red flows to allow to Consumer KP (top) and Producer KP (down) execute secure actuations in KUBIK

As shown in Figure 85 low part, the message flows are configured to allow SMOOL Producer KPs to send the specific actions to the hub that trigger the raising and lowering of the living room/kitchen blinds.

Figure 86 depicts an excerpt of the Java code of the Producer KP to gather sensor data and publish it to other SMOOL KPs (clients).

```
while (true) {
        if(!cs.isConnected())
        {
                cs = ss.accept();
        }
        System.out.println("Client connected");
        b = new BufferedReader(new InputStreamReader(cs.getInputStream()));
        while ((msg = b.readLine()) != null) {

                // --------------------SEND DATA-----------------------------------------
                timestamp = Long.toString(System.currentTimeMillis());
                System.out.println("Producing data at " + timestamp);
                String[] keyValue = msg.split(":");
                if (keyValue.length > 1)
                {
                        String key = keyValue[0];
                        Double value = Double.parseDouble(keyValue[1]);

                        switch (keyValue[0]) {
                        case "pirP0M1": {
                                pirP0M1 = (keyValue[1]).equals("1") ? true : false;
                                pirInfoP0M1.setStatus(pirP0M1).setTimestamp(timestamp);
                                producer.updatePresenceSensor(pirSensorP0M1._getIndividualID(), name, vendor, null, null,
                                                pirInfoP0M1);
                                System.out.println("Producing pirP0M1  " + pirP0M1);
                                break;
                        }
                        case "tempP0M1": {
                                tempP0M1 = Double.parseDouble(keyValue[1]);
                                tempInfoP0M1.setValue(tempP0M1).setTimestamp(timestamp);
                                producer.updateTemperatureSensor(tempSensorP0M1._getIndividualID(), name, vendor, null, null,
                                                tempInfoP0M1);

                                System.out.println("Producing tempP0M1  " + tempP0M1);
                                break;
                        }
                        case "humidityP0M1": {
                                humidityP0M1 = Double.parseDouble(keyValue[1]);
                                humidityInfoP0M1.setValue(humidityP0M1).setTimestamp(timestamp);
                                producer.updateHumiditySensor(humiditySensorP0M1._getIndividualID(), name, vendor, null,
                                                humidityInfoP0M1, null);
                                System.out.println("Producing humidityP0M1  " + humidityP0M1);
                                break;
```

Figure 86: Extract of Java code of the Producer KP

Once SMOOL messages containing sensor data (from the Producer KP) arrive to the Consumer KP, it sends orders to the actuators based on decisions over the values of acquired sensor data. To do so, two different Java classes are coded: 1) the Consumer main class which makes the direct connection with SMOOL server; 2) the Actuation class which decides the required actions to perform depending of acquired data. Figure 87 and Figure 88 show the extract of code of the described Java classes.

```java
public class ConsumerMain {
        public static final String name = "EnactKubikConsumer" + System.currentTimeMillis() % 10000;
        public static CustomActuation blindLivingUpActuation;
        public static CustomActuation blindLivingDownActuation;
        public static CustomActuation blindKitchenUpActuation;
        public static CustomActuation blindKitchenDownActuation;
        private static ServerSocket ss;
        private static Socket cs;

        public ConsumerMain(String sib, String addr, int port,int tcpPort ) throws Exception {

                SmoolKP.setKPName(name);
                System.out.println("*** " + name + " ***");

                // ------------PREPARE ACTUATION OBJECT--------------------
                blindLivingUpActuation =new CustomActuation(name,"blindLiving","Up");
                blindLivingDownActuation =new CustomActuation(name,"blindLiving","Down");
                blindKitchenUpActuation =new CustomActuation(name,"blindKitchen","Up");
                blindKitchenDownActuation =new CustomActuation(name,"blindKitchen","Down");

                // ------------CONNECT TO SMOOL----------------------------
                // SmoolKP.connect();
                SmoolKP.connect(sib, addr, port);

                // ------------SUBSCRIBE TO DATA---------------------------
                //Consumer consumer = SmoolKP.getConsumer();
                //consumer.subscribeToTemperatureSensor(new TemperatureSensorSubscription(createTemperatureObserver()), null);

                //-------------SEND ACTUATION------------------------------
                String timestamp = Long.toString(System.currentTimeMillis());

                ss = new ServerSocket(tcpPort);
                cs = new Socket();
                 // maximum interval that at least one message should arrive


                String msg;
                BufferedReader b = null;
                while (true) {

                        if(!cs.isConnected())
                        {
                                cs = ss.accept();
```

Figure 87: Extract of Java code of the Consumer KP.

```java
        public CustomActuation(String kpName,String name, String act) {
                this.kpName = kpName;
                //this.name = kpName + "_blindActuator";
                this.name=name;
                this.act=act;
                actuator = new BlindPositionActuator(name + "_actuator");
                blindPos = new ContinuousInformation(name+ "_blindPosition" );
                sec = new SecurityAuthorization(kpName + "_blindActuator_security");
        }

        public synchronized void run(double temp) {
                try {
                        System.out.println("Sending ACTUATION order for " + name);

                        // create the security metadata
                        String token = HTTPS.post(SECURITY_ENDPOINT, "{\"sub\":\""+kpName+"\",\"obj\": \""+ name +"\", \"act\": \" "+ act+"\"}")
                        sec=new SecurityAuthorization(kpName + "_blindActuator_security"+Integer.toString(counter++));
                        sec.setType("JWT + CASBIN payload").setData(token).setTimestamp(Long.toString(System.currentTimeMillis()));

                        // create the blindPosition information
                        blindPos.setValue(val++).setSecurityData(sec).setTimestamp(Long.toString(System.currentTimeMillis()));
                        blindPos.setDataID(name+act);

                        actuator.setBlindPos(blindPos);
                        // send to SMOOL
                        if (isFirstActuation) {

                                SmoolKP.getProducer().createBlindPositionActuator(name+act, kpName, "TECNALIA", null, blindPos, null);
                                isFirstActuation=false;
                        }else {
                                SmoolKP.getProducer().updateBlindPositionActuator(name+act, kpName, "TECNALIA", null, blindPos, null);
                        }
                } catch (Exception e) {
                        System.err.println("Error: the actuation order cannot be sent. " + e.getMessage());
                }
        }
```

Figure 88: Extract of Java code of the Consumer KP to generate and send a secure actuation order.

2.3.2.2.2 DevOps scenario for Security and Privacy Monitoring enabler

Being an Ops phase enabler, the S&P Monitoring enabler was integrated in the use case through deploying the needed security monitoring agents capturing the network data and the device data. The application data was captured through the message monitoring in SMOOL middleware.
The S&P Monitoring enabler back-end was installed at Tecnalia Cloud and configured specifically so as the network protocol analyser, the signature-based intrusion detection and the machine learning-based anomaly detection could work with the infrastructure of the Kubik, i.e., so as the tool was able to analyse data from the SCADA, the Raspberry PI, SMOOL, etc. in Kubik architecture shown in Figure 79. The loupe icons in the figure represent monitoring agents capturing the data and sending it to the enabler back-end for its processing.

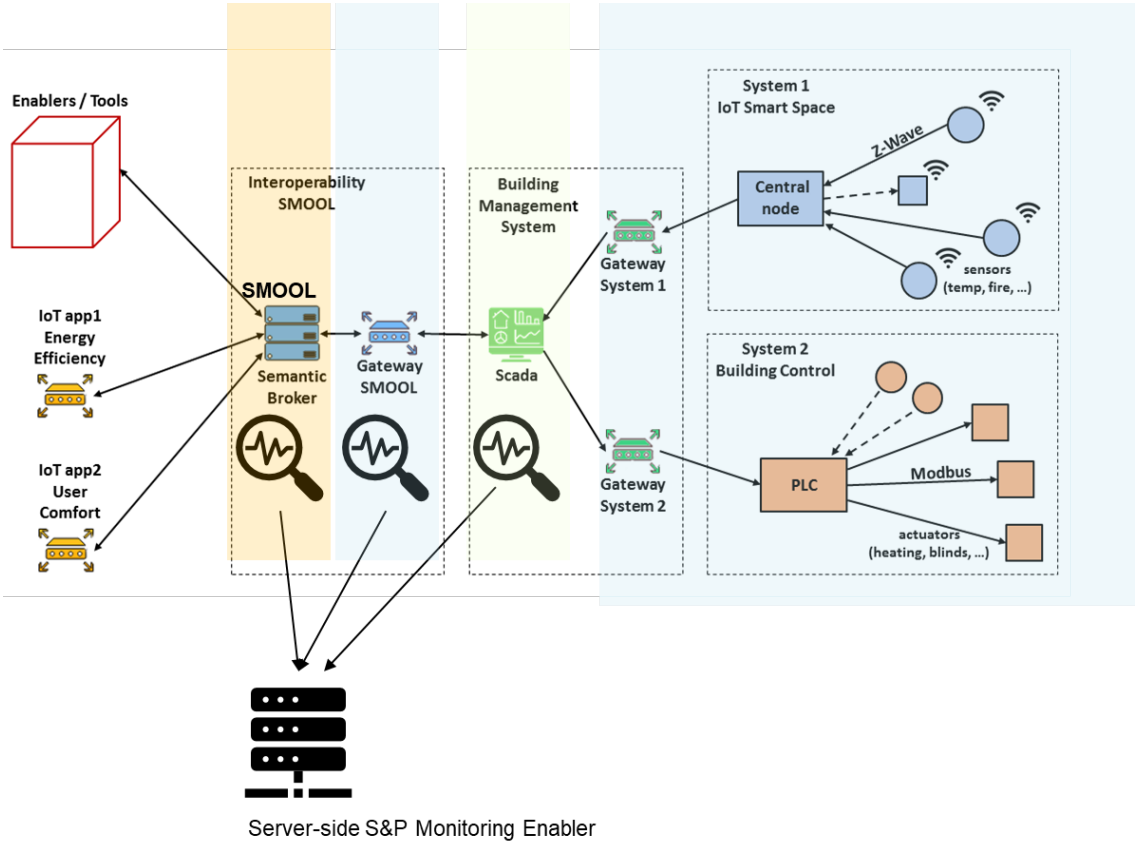

Figure 89: Security and Privacy Monitoring enabler integrated in Kubik.

In a second loop of the DevOps cycle the tool was enhanced with asset discovery features and refined anomaly detection. The asset discovery enabled to identify the assets (and their attributes) connected in the system and populate the asset inventory at start of SIS system and it was updated periodically every day (configurable). Passively the tool also analysed communications to discover new assets or changes in IP – MAC relations. The enabler was also able to analyse in the background whether any new asset was connected, get asset information and save it in the asset inventory.
The enhancements on the AI-based techniques enabled the detection of attacks by analysis of discrepancies with respect to normal behaviour. Attacks such as ARP spoofing or SSH user authentication brute force attack were successfully tested.
Figure 90 and Figure 91 show the Dashboard of the S&P monitoring tool while monitoring the traffic and devices in Kubik.

All the details of tool capabilities are fully explained in deliverable D4.3 Trustworthiness mechanisms for Smart IoT Systems - Final version of ENACT.

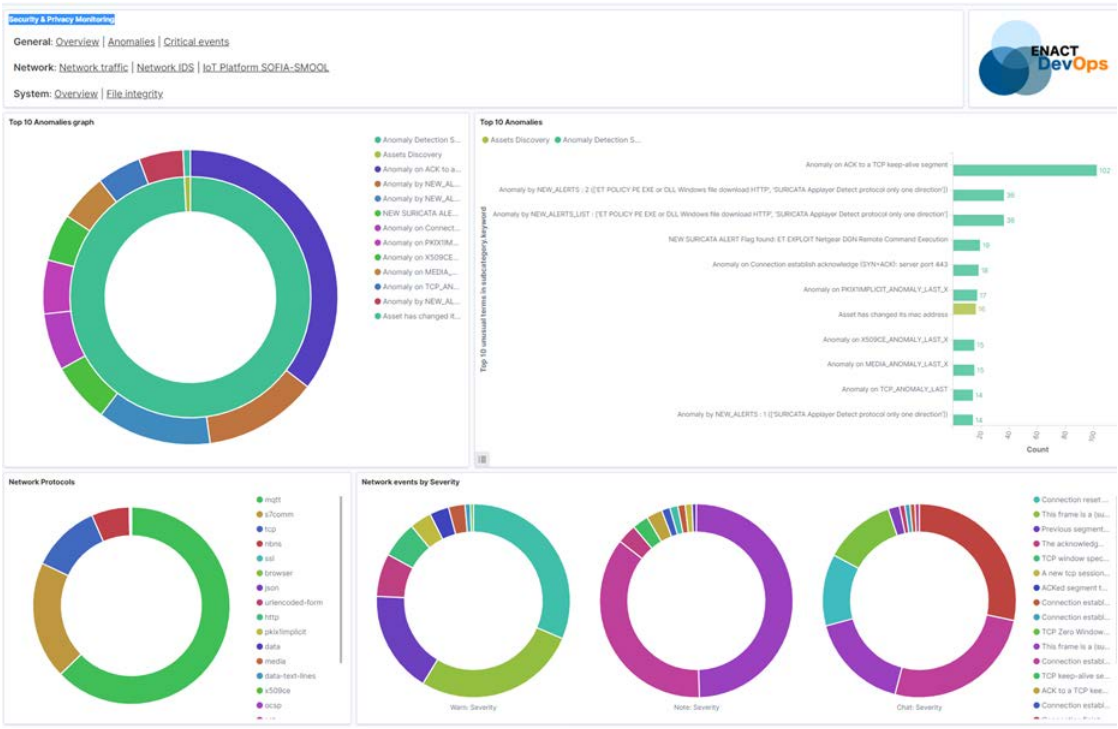

Figure 90: Dashboard (a portion) of S&P enabler integrated in Kubik.

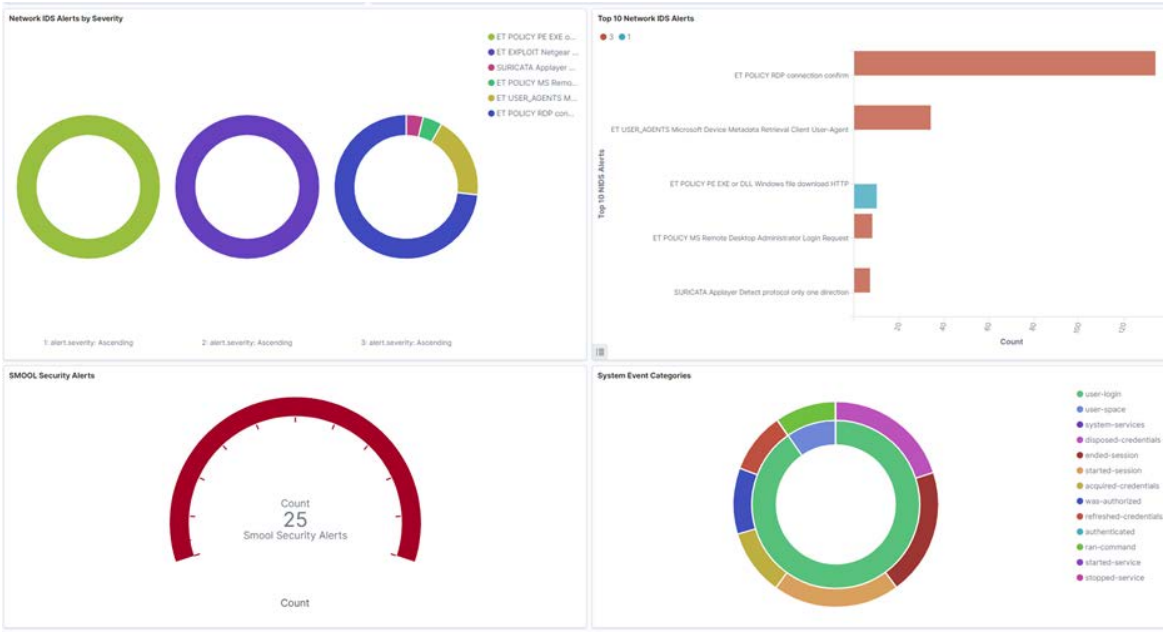

Figure 91: Smool security alerts (left down) in the Dashboard of S&P enabler integrated in Kubik.

2.3.3 Evaluation and Validation of the Business Case

Business KPI		Status
Main KPI		
Tecnalia is able to exploit Kubik as an infrastructure/testbed for the IoT domain (and not material)		DONE. The Kubik building now offers enhanced infrastructure sharing capabilities since the ENACT enablers ensure no actuation conflicts and security (secure communications, access control, threat detection, etc.) for all the IoT applications using the environment at the same time. This raises Tecnalia qualification as a provider for smart building lab services where clients can build and test trustworthy SIS.
Sub-KPIs (required to achieve main KPI)		
Provide customers with facility to run SIS	Increase of the number of actuators and sensors	Tecnalia has tested more than 20 new Z-Wave IoT sensors in the Kubik building (ground floor) of 8 different types and some IoT actuators such as wireless remote socket devices, roller blinds controllers and energy consumption meters in blinds and lights.
	By leveraging SMOOL communication and management of devices, customer effort to design SIS for the testbed is estimated to be reduced by %50	The customer effort to design SIS for the testbed is estimated to be reduced in half because the integration of IoT devices that use different protocols and legacy systems is done transparently using SMOOL communication middleware and not through protocol translation gateways.
	Customers applications development time is estimated to be reduced by 50% thanks to secure communications and security policies control by SMOOL	The development time of security code of IoT applications is estimated to be reduced in more than half the time because SMOOL KPs and server are now prepared to control secure communications and offer basic, customised or external security policy checking, as required.
Provide customers with secure and private environment	Number of security attacks detected improved by 100%	Before the S&P Monitoring enabler was integrated it was not possible to detect threats at operation, because there was no traffic monitoring nor intrusion detection available in the Kubik. Now all data from the SCADA, RaspberryPIs, etc. can be monitored by using S&P M&C enabler sensors, and it is possible to detect for example unauthorized access to resources, availability flaws or abnormal network traffic. The continuous monitoring of central parts such as the SCADA enables the situational awareness for all the SIS in the Kubik environment at a time.

	Protection from identified security threats at design time covers 100% of threats.	The S&P M&C enabler protects from the 100% of relevant security threats identified during SIS analysis, including the assurance of secure communications, the integrity of sensing and actuating data and the authentication of orders on actuators.
	Effort required to secure IoT app reduced by 90% through combination of GeneSIS and S&P control by SMOOL.	The effort to secure IoT apps is reduced by 90% through the use of GeneSIS to continuous deploy security mechanisms and KPs, and the use of security checking on the security concepts of SMOOL messages.
Provide the ability to run several applications in parallel	Enable the possibility to customize security for each app.	Now it is possible to secure all apps that are running in parallel in the IoT environment, but respecting the needs of each of them, through enabling the verification of the values in the security metadata of SMOOL KPs messages, including: authentication, authorisation, confidentiality, integrity and non-repudiation.
	Ability to manage concurrent access to actuators depending on the apps. Estimated effort reduced by 95%	The effort to manage concurrent access to the same actuators was reduced by more than 95% because the Actuation Conflict Management tools provided out of the box solutions in most to the cases.

Table 9: Smart Building - Business KPIs

The Kubik building has been traditionally used as an experimental infrastructure for developing and validating innovative products and systems to optimize energy efficiency in buildings, this ranged from passive systems such as modular insulating components for roofs and facades, to energy generation based on renewable energy and climate control systems. The ENACT project has leverage the Kubik building to develop new applications for energy efficiency and user comfort for Smart Buildings that are based on IoT equipment. This exploitation of Kubik as an infrastructure / testbed for the IoT domain is completely new, so most of the necessary features to be able to offer such activity did not exist before and the improvement made due to the ENACT project is very high.

2.3.4 Conclusions

Four sets of ENACT enablers were integrated and tested on the Smart Building case including KPI validation of the different enablers and execution of different validation scenarios, demonstrated the strengths and benefit of the ENACT tools for the use case. The main conclusions for the Smart Building use case are listed below:

- The Orchestration and Continuous and Deployment enabler (GeneSIS) is also used by most of the ENACT tools such as the Actuation Conflict Manager (ACM) enabler and is able to make deployments on of IoT applications in the Smart Building environment.
- The semantic middleware SMOOL platform (SOFIA) is fully integrated with ThingML and GeneSIS models and it is used as communication middleware in the Smart Building use case so as to continuously check and ensure trustworthy smart things communicate in the environment and security policies are enforced at message layer, e.g. requiring messages to include authentication tokens.
- The Actuation Conflict Manager (ACM) enabler is fully operational and resolve the sending of conflicting orders by different applications to the same actuator or to different actuators acting of the same physical variable.
- The Security and Privacy Monitoring enabler actively monitors continuously the network traffic and device data so as to identify whether the traffic deviates from normal behaviour, malicious

actors try to enter the system or unauthorised entities try to access resources without permissions.

- The Online Learning enabler is able to improve comfort and cost of a Smart Building controller.
- The Risk-Driven decision support enabler is able to propose a set of mitigation actions against the asset that direct technical requirements from the trustworthiness perspective.

3 Overall Evaluation of the Enablers

This section will provide a general conclusion of the evaluation of the performance of each enabler after their validation in more than one use case.

3.1 Orchestration and Continuous Deployment enabler (GeneSIS)

This enabler, also known as GeneSIS, supports the automatic deployment of software, together with the attached security mechanisms, across the computing continuum from IoT devices to Edge to Cloud. Developers use a declarative modelling language to specify what software components and security mechanisms they want to deploy, and the engine automatically deploys them into the resources in the computing continuum, continuously monitoring the deployment status. The GeneSIS modelling language is independent of the underlying technologies, i.e., GeneSIS can deploy components anywhere in the IoT-Edge-Cloud continuum: from microcontrollers without direct Internet access to virtual machines running in the Cloud. It also includes security mechanisms as first-class modelling elements thus promoting security-by-design. More precisely, GeneSIS offers off-the-shelf components for deploying security solutions, mechanisms to inject security policies in the SMOOL IoT platform, platform-independent rolling deployment mechanisms for reliability and availability. The tool is essential for deploying systems running across IoT devices, Edge and Cloud resources.

3.1.1 Final evaluation summary

The GeneSIS tool has been evaluated in the three use cases, all illustrating different benefits and specificities of the tool, eventually facilitating their DevOps. The ITS use case focused on performing remote deployment while ensuring maximum availability. The Smart Building use case focused on the integration of GeneSIS within a complete DevOps environment, integrating many of the ENACT enablers, and running multiple adaptations in the deployment of the system. Furthermore, a specific focus was also on the deployment (i) of security control mechanisms and (ii) of software components on devices with no access to internet. The eHealth use case focused on the proper integration of DivEnact and GeneSIS. DivEnact focusing on the management of the fleet of gateways and GeneSIS providing means for last mile deployment including on devices locally connected to the gateway. In this use case, integration with ThingML and providing means to rollback deployment on failure were also aspects of importance.

The evaluation proves the following benefits:

- DevOps teams can use the same tool and language to deploy software on infrastructure ranging from small devices with no internet connection to powerful Cloud infrastructure. This relieves them from using several tools for each type of infrastructure (e.g., Arduino CLI for Arduino devices, Ansible for Gateways, ThingML CLI for cross-platform code generation, Cloudify for cloud resource provisioning, etc.). Moreover, the single abstraction offered by the GeneSIS language also relieves developers from maintaining in synchrony multiple deployment scripts for each tool, which is typically required to ensure the proper integration of the SIS once deployed. This has been demonstrated in all use cases.
- Developers can, by design, integrate security solutions and policies as part of their deployment. Facilitating the addition and evolution of security solutions over time and limiting the impact of such evolutions on the development of the applications. This has been demonstrated in the context of the smart building use case.
- DevOps teams can operate SIS in a more effective way. They are provided with a single-entry point (a GeneSIS deployment model) to specify how to adapt a deployment in a declarative way. How to enact the adaptation is computed by GeneSIS with the objective to maximize availability, on adapting the part of the system that needs to be. This has been demonstrated more specifically in the context of the Smart Building use case.

- The implementation of high availability and reliability solutions by DevOps teams is facilitated. GeneSIS provides means to select if software components will be automatically deployed using software replication and rolling deployment strategies (e.g., blue/green deployment), or not. This provides a seamless way to implement availability and resilience solutions into the deployed SIS (i.e., multiple replicas of a software, automatic restart and re-routing of requests when a replica fail, rollback to old version when new version is not performing as expected). Following the first bullet point in this list, the availability and reliability mechanisms can be specified in a platform-independent and/or specific way, making them easily applicable to many infrastructures. These points have been demonstrated both in the context of the eHealth and ITS use case.

From a non-functional point of view, the evaluation confirms that the GeneSIS enabler is easy to be integrated with other DevOps tools, both the ones within ENACT and the third-party ones. In particular, the integration of GeneSIS with Risk Management, ACM, BDA, Test and Simulation, security and privacy monitoring, DivEnact enablers has been demonstrated. The integration with third-party tools has been done smoothly with mainstream DevOps solutions such Jenkins, Docker and Ansible. In term of scalability, even if the tool is mainly meant for deploying a single system (by contrast with DivEnact, which focuses on the deployment of systems of systems and delegates the deployment of a single system to GeneSIS), deployment of a single system involving 31 software components, 1 cloud resource, 6 edge nodes, and 51 devices has been demonstrated in the context of the smart building use case. Finally, integration with all major IoT platforms has been demonstrated (TelluCloud when deploying the eHealth use case, SOFIA (SMOOL) in the Smart Building use case, and FIWARE in the ITS use case).

3.1.2 Recommendations

The application of GeneSIS into the use case DevOps practice confirmed the practical relevance of the tool. In particular the tool has proven to be a cornerstone in many DevOps pipeline, facilitating the integration of several of the other enablers. The complementary between GeneSIS and DivEnact is natural and highly relevant in many contexts, thus we believe that augmenting the integration between GeneSIS and DivEnact into a single deployment solution for Edge and IoT-based systems will be key for both tools in the future. This will be further investigated in a new project funded by the Research Council of Norway, with SINTEF, Tellu and another SME from Norway.

3.2 Test, Emulation and Simulation enabler

The Test and Simulation enabler provides a set of features to help IoT developers easily test their IoT system. The tool supports two method to build the testing datasets. The first method is recording data from a real system which provides the real data for testing. The second method is using the regular & malicious data generator to generate datasets based on the behaviour of the sensors, this method helps to build datasets to cover many testing scenarios which are very difficult to produce in reality. With the prepared datasets, the Test and Simulation tool supports to build up an automation testing process, easily integrate into any DevOps cycle. The tool can involve into the IoT system development from early state to every deployment of the system.

3.2.1 Final evaluation summary

The Test and Simulation enabler (TaS) has been evaluated in three use cases to show different features of the tool and has proved an important role in their DevOps.

In the ITS use case, the tool has been used in the focus of testing the scalability of the ITS Gateway. The benefits that the tool has brought to the use case:

- Record the data of a single train and store the recorded data as a dataset for testing.
- Use the recorded dataset to simulate a single train and measure the reaction of the ITS Gateway.
- Easily scale up the number of simulated trains to 5, 10, 20 and 50 and measure the maximum capacity of the ITS Gateway.

The evaluation of ITS use case has confirmed that by using the tool, the scalability testing can be done very easily and with very low cost of buying and installing the testing environment.

In the Smart Home use case, the has been integrated completely in their DevOps cycle and show many benefits:

- Record the data of the real system and store the recorded data as testing datasets.
- Generate testing datasets to cover a large number of testing scenarios which are very difficult to produce with the physical sensors network. The tool has generated data based on many different behaviours of the sensor, from normal behaviour to abnormal behaviours such as: invalid value, value out of range, etc., and also the tool has generated data based on several cyber-security attacks such as: DOS attack, Slow-DOS attack, node failed.
- Automation testing: by setting up the test cases, test campaigns, all the testing scenarios can be tested automatically with a single API call.
- Interact with GeneSIS: everytime GeneSIS deploys new model of the system, it also makes a request to a special end-point defined by Test and Simulation. The trigger launches all the testing scenarios automatically, then the results will be sent back to GeneSIS, so the GeneSIS can have a proper reaction on the testing results.

The Smart Home use case has validated that the Test and Simulation tool can be easily integrated into any DevOps cycle, the time of development has reduced by using automation testing, and many testing scenarios have been tested by using the prepared datasets.

In the eHealth use case, the Test and Simulation has been used to simulate 100 Safety Alarms to test the scalability of the Gateway. It also simulated a cyber security attack where the attacker sends invalid value data to check if the Gateway can handle it. Moreover, by using the tool to simulate several sensors, the use case has proven that the tool can be used to test how well the sensor bridge and TelluCloud Agent handle the abnormal data format.

3.2.2 Recommendations

The tool has proven to bring many benefits in three validated use cases. In the future, Montimage has planned to apply the tool in more use cases (in several EU research projects). Also, there are still some more improvements and features of the tool can be implemented in the future:

- For the scalability testing, sometime a single instance of Test and Simulation enabler can have some limited due to the capacity of the hosted machine. Multiple instances of Test and Simulation enabler deployed on demand will open a huge potential of the tool to test the system at any scale.
- The tool has been inspired by Digital Twins concept, however there is still missing the prediction process. By using the live recorded data from a real system and forward the recorded data to a testing system, the tool then adds some manipulations on the input test many difference simulations at the same time, provides the output of the prediction for many scenarios thus helps the IoT operator to have better decision.

3.3 Online Learning enabler

The Online Learning enabler offers means to apply Reinforcement Learning (RL) machine learning techniques to control problems whose underlying structure can be formulated as a RL problem. Thereby a tool is provided which relies on publicly available implementations of two popular RL algorithms, with an interface enabling to interconnect these algorithms with an arbitrary system via MQTT or HTTP. The tool has a graphical user interface (GUI) to enable a DevOps engineer to properly configure the tool and to provide several means to monitor the learning process. Furthermore, APIs are provided to interconnect the tool with other tools of the ENACT Framework.

3.3.1 Final evaluation summary

The Online Learning enabler has been validated in the Smart Building use case. Therefore, a RL problem has been formulated which basically comprises the process of finding a strategy to control an HVAC system in a certain room of the KUBIK building, that maximizes user comfort and minimizes energy consumption. This RL-problem is solved by means of a certain RL solution method: policy-based RL (with PPO serving as an algorithm candidate). A simple thermostat implementation thereby served as a baseline. As it was not feasible to demonstrate the capabilities of the Online Learning enabler by applying it to the physical facilities of TECNALIA without blocking these for an undefined amount of time while the facilities are used to run several other experiments in parallel (that would influence our experiments), we decided to set up a simulation based on the specification of the physical facilities. The simulation is written in Python programming language using the OpenAI Gym framework and can be found here: https://gitlab.com/enact/online-learning-enabler/-/tree/master/HVAC-Simulation. A detailed description of the simulation can be found in D3.3.

Using the simulation is has been shown that the OLE has a positive impact on user comfort and energy consumption compared to the baseline thermostat approach:

	Reward	**Reward (250.000)**	**Reward (450.000)**	**Energy**	**Energy (250.000)**	**Energy (450.000)**	**TC**	**TC (250.000)**	**TC (450.000)**
Baseline	27006777	14527144	3432345	-1777270	-801690	-342090	28784047	15328834	3774435
RL	-5,71%	12,43%	29,72%	55,48%	33,36%	61,12%	-8,79%	10,04%	21,49%
RL (with initialization)	17,11%	15,30%	30,74%	48,25%	35,89%	63,21%	13,07%	12,62%	22,23%

As can be seen in the table above, the RL approach (RL) outperformed the baseline approach in terms of total reward, energy consumption and thermal comfort after it has been converged (250.000). In the overall comparison it performs worse than the baseline approach due to its initial learning phase. To improve the performance of the RL approach it has been initialized through a systematic pretraining approach (cf. D3.3). With the RL approach being initialized (RL (with initialization)) it outperforms the baseline approach in all metrics over the whole experiment duration. Further details concerning the evaluation can be found in D3.3.

The evaluation experiments have shown that is possible to use the OLE to learn a control strategy for a HVAC system that outperforms a simple baseline thermostat approach. Furthermore, it has been shown that the OLE can be successfully be deployed with an orchestration tool (such as GeneSIS) and can be interconnected with other tools (such as BDA).

3.3.2 Recommendations

After applying the OLE to a simulation of the KUBIK building, a next step would be the application in the real KUBIK building, especially to further investigate the impact of a systematic initialization of the RL approach to improve the online learning performance.

3.4 Root Cause Analysis enabler

Root Cause Analysis's (RCA) aims to infer the root-causes of problems by analysing the causal chains governing the system under monitoring. In principle, the RCA enabler relies on machine learning algorithms to identify the most probable cause(s) of detected anomalies based on the knowledge of similar observed ones. It consists of the construction of a systemized historical database of known/learned incidents together with their corresponding symptoms, root-causes, impacts and

mitigation actions, as well as the calculation of the similarity between the symptoms of new incidents with the ones stored in the database. It requires, thus, enough relevant monitoring data attributes and significant domain and system knowledge for the efficiency and accuracy of the analysis. RCA aids the monitoring task of the system's administrators and the DevOps engineers, so that they have more hints to respond quickly to the incidents.

3.4.1 Final evaluation summary

The RCA enabler has been validated in the ITS use case in which it collects data concerning the hardware usage (e.g., CPU, RAM, power consumption, disk usage) and connectivity status (e.g., number of gateway connections, number of gateway up connections, published message size, received message size, number of messages published per second, number of messages received per second, and messages delay) and visualizes them by the charts on dedicated dashboards.

The RCA tool works in basically two phases:

- Knowledge acquisition phase: RCA first learns the images of the system reflected in aforementioned monitoring data when it is properly functioning. It is then possible to actively inject the failures/ problems with known causes to the ITS so that those incidents' patterns can be extracted and learned by the RCA. This leads to the construction of a historical database of proper functioning events and observed incidents with known causes, impacts and mitigation actions to be taken.
- Monitoring phase: RCA is monitoring the system in real time and calculating the similarity score of the actual state and each of the learned events (normal states and incidents) in the historical data. In consequence, when the system is seen too different from observed normal states, the administrator/ DevOps engineer can be notified so that the potential current abnormal behaviour can be learned and updated to the historical data. Similarly, the repetition of a failure/ problem reflected by a high score can be alerted and corresponding root-causes, impacts and mitigation actions can be suggested to react quickly to the incident.

In summary, the following results have been observed:

- RCA recognises correctly when the system is under failures and when it is in a properly functioning mode, as demonstrated in section 3.2.1.3. When a failure occurs, RCA see it less than 24% similar to the learned normal states.
- The similarity score between the learned failures and their reoccurrences are in general higher than 0.8 (80%)
- When the system is properly functioning, it looks more than 85% similar to the learned proper states.

3.4.2 Recommendations

The ITS is still in integration phase and has not been fully deployed yet. In the production phase, there might be a more complex system with a bigger number of devices to be monitored as well as of the potential failures that could occur. Indeed, more tests need to be performed in the future to verify the scalability and the performance of the RCA in monitoring such system.

3.5 Actuation Conflict Management enabler

The Actuation Conflict Manager enabler supports the automatic detection, analysis and resolution of direct and indirect actuation conflicts. The enabler relies on the WIMAC model (Workflow and Interaction Model for Actuation Conflict management) built upon the deployment model of the SIS (provided by GeneSIS) and an environment model that links actuators to the physical properties they act on (provided by designers). Thus, the overall conflict management process applies over an abstract representation of the SIS that is decoupled from its detailed code. The tool is conceived to (i) automate

as much as possible conflicts identification and management processes by proposing ready-to use safe actuation conflict management solutions and (ii) integrate with other continuous delivery and deployment solutions for maximum agility at the heart of the DevOps life cycle. Moreover, state-of-the-art verification techniques ensure that the conflict management solutions injected in the SIS satisfy some temporal and logical properties making DevOps teams confident in deploying the new version in the system.

3.5.1 Final evaluation summary

The Actuation Conflict Manager tool has been evaluated on three use cases: ITS, Smart Building and Smart Home, all illustrating benefits and specificities of the tool, also facilitating the development of the use cases. The ITS use case focused on detecting and solving direct and indirect conflicts in a simulated environment. This results from the fact that it was not possible to create actuation conflicts potentially harmful in the realm of a real train. The Smart Building and Smart Home use cases focused on applying the Actuation Conflict Manager tool to large scale SIS operating in real physical environment. For instance, the Smart Home system consists of more than fifty applications and more than two thousand node-red nodes plus ThingML code.

The evaluation proves the following benefits:

- DevOps team can use the tool to locally detect and solve direct and indirect actuation conflicts without having to handle this problem at the code level (often in an ad-hoc fashion). The overall actuation conflict management process applies over an abstract representation of the SIS that is decoupled from its implementation.
- The developers can automate as much as possible the actuation conflict management process and integrate with other continuous delivery and deployment solutions.
- This tool was designed to be integrated into an agile and continuous delivery process, thus meeting the criteria of incremental and agile development advocated by the DevOps approach. By proposing safe ready-to-use off-the-shelf actuation conflict management solutions, the tool reduces development time yet maintaining a high-quality level.
- In the case where the conflict management is more complex and cannot be satisfied with off-the-shelf solutions, the tool allows designers to develop custom actuation conflict management solutions. State-of-the-art verification techniques then ensure that the custom solution to be injected in the SIS satisfies some temporal and logical properties making DevOps teams confident in deploying the custom solutions into the system. The logic of a custom solution is specified on the basis of a language close to ECA rules (Even - Condition - Action). This formalism has been selected to reduce the learning curve, as being intensively used in Smart Home and Smart Building approaches (like IFTT for instance).

3.5.2 Recommendations

The use of the enabler at the heart of the DevOps approach for the development of the different use-cases confirmed the practical relevance of the tool. In particular, the tool has proven to be a key component to detect and solve actuation conflicts in complex large-scale systems (with multiple applications interacting concurrently in the physical environment). The complementary of this tool with the other tools used at Dev time was particularly appreciated as it prevents deploying conflicting applications potentially leading safety-critical problems in the physical environment.

3.6 Context and Behavioural Drift Analysis enabler

The Context and Behavioural Drift Analysis enabler supports the quantitative effectiveness assessment of SIS at run time and, at design time, the analysis of its drifts. The effectiveness assessment relies on a model of the SIS legitimate behaviour, i.e., the legitimate effects the SIS has to produce in the physical environment in different contexts, fully decoupled from its implementation. The behavioural drift analysis tool then, is conceived to provide designers with the symptoms of the drifts in effectiveness beyond the quantitative evaluation. The process is fully automated. It first learns a behavioural model

of the SIS from the real-world observations that led to the drifts. Then, it compares this model with the model of the SIS legitimate behaviour and generates a dissimilarity graph that makes clear the structural and parametric differences between both models. Both tools then cover a full DevOps lifecycle, first ensuring that the SIS operates as expected at run-time and then helping designers understand its drifts in effectiveness.

3.6.1 Final evaluation summary

The Context and Behavioural Drift Analysis enabler has been successfully evaluated on ITS, Smart-building and Smart-home use cases. For obvious security reasons, it was not possible to cause a drift in effectiveness in the case of ITS. Instead, a simulator was used (OpenBVE) to simulate several unexpected behaviours, all of which led to behavioural drifts that the enabler was able to detect while identifying their symptoms. In the Smart-home and Smart-building use cases, several models of legitimate behaviour have been developed, covering several properties of the physical environment such as the luminosity, the sound, etc. In general, the models were meant to cover the range of actions that can be triggered by the SIS, ensuring that all have an observable impact on the physical environment.

The enabler brings numerous advantages:

- It provides an objective assessment of the effectiveness of actions triggered in the physical environment. It is no longer a question of considering that the sequence of orders is correct or not (PASS/FAIL) but also, considering the uncertainties pertaining the physical environment, of ensuring that they have the desired effects (quantitative gradual assessment where 0 means that the behaviour observed is completely unexpected and 1 means that SIS behaves as expected).
- When the SIS does not behave as expected, the analysis part allows to compare the observed behaviour with the legitimate one and thus, to identify the underlying symptoms. This not only allowed us to detect problems deliberately caused in the physical environment to make the SIS drift, but also to identify problems that were not anticipated, thus proving its effectiveness. For instance, in the Smart-home use-case, a drift was expected to be produced by a smart vacuum cleaner. While the drift effectively occurred, another one, completely unexpected, was caused by the dishwasher, programmed to start at a particular time.

3.6.2 Recommendations

Legitimate behaviour models are a bit difficult to develop manually. However, the automatic model learning capability offered by the drift analysis tool, when applied to expected observations, provides a first model that can be used as a basis for further enrichments.
Linking the causes of drifts in effectiveness to the symptoms identified by the analysis tool requires deep knowledge in the SIS development and its operating (physical) environment. Thus, deep human analysis is still needed for now.

3.7 Diversifier enabler (Tool name: DivEnact)

IoT diversification enabler was designed to maintain the diversity of software instances deployed in a large number of IoT/Edge devices. We developed this original design into a novel concept: fleet deployment, to allow DevOps teams to deploy multiple software varieties into the device fleet as a whole, instead of handling all devices individually. The fleet deployment concept keeps the essential features of the diversifier, i.e., maintaining the diversity of the software among the devices, supporting the recovery and rollback of software on devices without handling the devices independently, etc. To realize these features, we implement the fundamental functionalities involving the monitoring of devices and their contexts within the fleet, assigning proper software variants to the devices depending on their contexts, and automate the "last-mile" deployment of software components on the edge and IoT devices. The tool is essential for the DevOps of IoT applications in the production phase, when the application is running on a large number of gateways at the end-users' side.

3.7.1 Final evaluation summary

The DivEnact tool is evaluated on the Tellu use case, facilitating the DevOps practice of the Tellu development team. Among the three use cases, Tellu's eHealth use case is the one that is already in the production phase, with 200+ gateways already used by the end-users, i.e., elderly people living at home in the hospital. Together with Tellu, we have built a proof-of-concept DevOps environment integrating DivEnact, GeneSIS and CAAC tools, together with external DevOps tools already used by Tellu. We have conducted sample DevOps activities based on the scenario described in Section 3.2.2, to evaluate the effectiveness of using the fleet deployment concept and the DivEnact tool to support DevOps practices.

The evaluation proves the following benefit that DivEnact brings to Tellu's DevOps team:

1. Developers can maintain multiple diverse software versions among their development and production devices, in an automatic and more effective way. Under the fleet deployment concept, developers deploy software variants into the abstract fleet, without having to individually maintain the individual devices. In the sample scenario with 20 gateways and 5 software variants, this improves the productivity by approximately 3 times: instead of manually checking the context of the devices, assigning the proper software variant to each device, and launch batch deployment script independently, DivEnact allows developers to deploy all the variants in one shot, and the DivEnact enabler will automatically assign the variant and launch the deployment.
2. Developers can operate the devices in a more effective way. DivEnact provide a global view of all the devices, their context and the software deployed on them. Developers can easily manipulate the devices, such as roll back deployments, or set up automatic operation tasks, such as recovering of devices to default deployments. These has been shown in the tests TO3.1 and TO3.2 in Section 3.2
3. Developers also obtain some fleet-level controls to the devices and their software, which are not supported in their previous deployment environment, such as mass publishing of a new release to the whole fleet, preview new variant on selected devices (users), roll back all devices deployed with a particular version, etc. These operations reveals the effectiveness and potential of the tool.

From a non-functional point of view, the evaluation confirms that the DivEnact enabler is easy to be integrated with other DevOps tools, both the ones within ENACT and the third-party ones. In terms of scalability, we tested the tool on an experimental fleet with 8 physical gateways: 3 of them are of the same type as used in Tellu's production, and the rest are Raspberry Pis. The gateways are distributed in different places: in Tellu, SINTEF, our home offices and one in ISRAA in Italy. Counting all the different sensors connected to all these gateways, we have in total 20 devices. With the help of Azure platform and the Test and Simulation enabler, we managed to enlarge the fleet into 200 devices, the majority of them are virtual gateways and simulated sensors.

3.7.2 Recommendations

The application of DivEnact into Tellu's DevOps practice confirmed the practical relevance of the fleet deployment concept, and also encourages us to consider other DevOps activities from a fleet point of view, such as the provisioning of devices, the monitoring of devices and applications, etc. We extend the concept from fleet deployment into Fleet Operation, and will further investigate it in a new project funded by the Research Council of Norway, with Tellu and another SME from Norway.

3.8 Risk Management enabler

ENACT Risk Management enabler is a tool to help organizations to detect vulnerabilities in their IoT Systems' architecture and predict the impact on the software development process. On the one hand, this enabler helps organizations to analyse the architecture of their Smart IoT Systems and detect potential vulnerabilities, thus supporting the selection of the right components in the system by detecting characteristics that may be important to mitigate those vulnerabilities. On the other hand, the enabler is

able to understand how vulnerabilities and potential associated risks may impact the software development process, for instance in terms of generating delays with respect to planned tasks and jeopardizing the ability of a company to continuous deliver according to the commitments with customers.

3.8.1 Final evaluation summary

Risk Management enabler was evaluated by two use cases: the eHealth use case from TellU and the smart building use case where Tecnalia evaluated the data flow and automatic vulnerability detection of the tool.

The eHealth use case of TellU showcases perfect use scenario for Risk Management tool. Not only the use case is based on DevOps cycle in IoT space, where the core business of the company is matching exactly to the project and enabler value proposition, but also the Risk Management is an obligatory step required by law in the health space. Due to these reasons, two case scenarios were evaluated with TellU. First one allowed to use the existing GeneSIS model of the TellU cloud and run a full run of the risk management scenario usually performed by the Risk Manager of the company. All of the capabilities required by the typical analysis were confirmed and the approach to risk management validated by the use case. Within the architectural scenario, the evaluation focused on the specific use case of continuously monitoring the risk of exceeding the daily limit of the SMS sent by the TellU Cloud though a set of third party providers. Wide range of integrations and mitigation actions were tested during the scenario.

Second scenario of evaluation of the eHealth use case focused on the automatic risk detection based on the data flow diagram showcasing the flow of data of the scenario described in the architectural analysis. This allowed for a benchmark of relevancy of detected vulnerabilities, risks and mitigation actions. The exact outcomes of this evaluation have been described in the section 2.2.2.2 of this deliverable.

The smart building use case of Tecnalia represented a scenario of the IoT based software evolution. In the case of the evaluation, a specific scenario from the KUBIK building has been taken on board where the main evaluation has been performed on analysing the completeness of the automatic vulnerability detection knowledge base.

3.8.2 Recommendations

The main recommendations from the use case pointed out has been around the additional commercial future required in order to fully integrate the tool within business setup of the respective use cases. The recommendations are matching closely to the business offer that Beawre represents to the commercial customers where the tool offered builds on top of the Enact Risk Management enabler.

3.9 Context Aware Access Control enabler

The Context Aware Access Control (CAAC) enabler is a solution for context-based dynamic authorization, which allows to control in the same way the access of all the IoT actors (end-users, services, devices, administrators) to the operated data and resources, for both IT and OT (operational technologies) domains. In particular, this tool provides Context-aware risk & trust-based dynamic authorization mechanisms ensuring *(i)* that an authenticated IoT node accesses only what it is authorized to and *(ii)* that an IoT node can only be accessed by authorized software components. Access authorizations are adapted according to a risk level computed from contextual data on the user and his devices. This tool can be used in all IoT use cases where access control to protected resources is required. It is of special interest if this access control must consider the context and the resulting risk level.

3.9.1 Final evaluation summary

The Context Aware Access Control enabler is evaluated on the TellU Digital Health use case. Among the three use cases, TellU's Digital Health use case is the one that is already in the production phase, with 200+ gateways already used by the end-users, i.e., elderly people living at home in the hospital.

The Context Aware Access Control is interesting in the Digital Health use case to achieve some more advanced functionality allowing access to features and edge and IoT devices based on context. For example, surveillance cameras in some elderly people's homes is normally only allowed to be accessed during scheduled time periods (planned supervision) to respect and adhere to privacy requirements. However, in case of emergency, e.g., when a safety alarm button is activated, the access to watch video from camera can be permitted even though it is outside the scheduled time periods for planned supervision. Thus, the context-awareness allows privacy rules to be overridden when, for example, the person is in danger. What happens is that if the alarm button is activated, the risk level associated with the user is lowered to grant more access to the devices belonging to this user. In general, we will be able to set up that the more critical the contextual information, the lower the risk, and the greater the access to resources.

The testing scenario is introducing a new access control mechanism into the RPM fleet. The RPM service monitors the real-time status of a patient, and it requires strong access control to protect the data privacy for the patients. Among the sensors, the cameras are used for the most sensitive data, i.e., the live video of the patients and their living environment.

To achieve this, TellU integrated the Context Aware Access Control enabler and applied it in TellU experimental tool chain for the ENACT eHealth case study. For each request to the device data, the TellU backend service invokes the CAAC service to decide whether the request should be authorized. The CAAC extends the traditional access control service, by considering the context of the data providing device, to evaluate a contextual risk level that is considered when making the authorization decision.

Evidian has given TellU access to a testing platform that provides CAAC in SaaS mode: https://enact-caac-my.evidian.com , as well as a user guide. This platform also contains a simple risk server, intended to validate the architecture and to provide risk levels depending on both the contextual events sent by the connected devices, and the audit event generated by Evidian Web Access Manager following the users' connections. This risk server is not part of the developments provided by Evidian for the ENACT project, but it is used to demonstrate the functioning of the CAAC tool and its integration into the future Cloud architecture of the Evidian "Prescriptive IAM" module. A mock-up application (https://enact-caac-riskserver-ui.evidian.com/) is provided to be able to tune the taking into account of the contextual events by the risk server.
The CAAC authorization server is available at: https://enact-caac-auth.evidian.com.
The User's Token Management UI is available at: https://enact-caac-my.evidian.com/oidc-provider/

The evaluation proves the following benefits:

- The scenario allows to demonstrate the control of access to the TellU gateway to authorized users only and to authorized services only.
- The CAAC is relevant in several scenarios related to the TellU services, both in terms of controlling access from a device on behalf of a user (e.g., a device or service are granted access based on context information) or for particular users to get access to a service based on the context.

3.9.2 Recommendations

The Context Aware Access Control enabler is an innovation that is very relevant for TellU in order to do authorization management at the Gateway level.

These use cases should generally be handled as part of TellU services and it would be advisable for TellU to consider further exploring the application of the CAAC enabler in future services.

3.10 Security and Privacy Monitoring enabler

The Security and Privacy Monitoring enabler was successfully tested to be a very useful tool to provide situational awareness of security issues and attacks in the use cases of ITS and the smart building where it was tested. The tool focused on protecting from data availability, integrity and confidentiality issues through raising alarms and showing all the needed information and details in a dashboard in the Cloud.

3.10.1 Final evaluation summary

The main function of the S&P Monitoring enabler in the ITS use case was to continuously monitor the MQTT communications of the Gateway (edge) onboard the ITS and enable the remote monitoring to the operator of the onboard system to supervise at operation the security status of the Gateway. The tool makes it possible to detect both abnormalities in the network traffic and intrusions of unauthorised users of the Gateway and notifying accordingly to the system in the Cloud. Therefore, we can say that the tool successfully fulfilled all the technical objectives stated for the use case with respect to security aspects.

Similarly, the tool was integrated within the Kubik Smart building to enable the monitoring of all the communications, including those of IT and OT industrial protocols. The advanced cyber threat detection capabilities of the tool served not only for ENACT pilot applications but also for other IoT applications running in the infrastructure, making Kubik more secure. A number of issues such as unexpected connections of IoT devices to external servers, abnormal device downtime, etc. together with tested attacks were detected and notified by the tool so as Kubik operators could correct them.

3.10.2 Recommendations

The AI-based detection capabilities of the tool are expected to be further improved and enhanced so as unknown attacks are detected too. The machine learning techniques would benefit from incrementing the data acquired by the tool so it is recommended that the tool is integrated with the SIS and that a training period is dedicated to the refinement of the attack detection capability. The longer the period the more data would be captured, and the efficiency of the detection would be improved.

3.11 Security and Privacy Control enabler

The main objective of the S&P Control enabler was to automatically react to issues detected by the monitoring service. According to the integration possibilities of the use cases, the control service was implemented as part of the SMOOL capabilities, and therefore integrated in the Smart Building use case that uses this middleware and where the tests on control actions were performed.

3.11.1 Final evaluation summary

The integration of the S&P Control enabler in the Smart Building use case enabled to enforce secure communications between the smart devices deployed in the Kubik infrastructure. The enabler was successfully tested to empower the SMOOL server and KPs improved in act to act as control mechanisms when the security metadata in the communications messages where not the appropriate ones. The use of security-enhanced SMOOL enables three different types of security architectures of SIS deployed in Kubik: i) basic security in the KPs for cutting communications when discrepancies in the metadata are detected, ii) ad-hoc customised security policies within the KPs so as different reactions (notifications, communication cut, requests for metadata encryption, etc.) are possible, and iii) security policy enforcement executed by external systems.

3.11.2 Recommendations

To fully benefit from the full capability of the S&P Control enabler, it is strongly recommended that the S&P Control enabler is used together with the S&P Monitoring enabler. The monitoring service is

prepared to show the notifications from SMOOL in a dedicated dashboard where all the details of the alerts raised by SMOOL can be shown. Another main recommendation is to use both S&P enablers together with SMOOL and GeneSIS. This would allow the DevOps team to understand the issues in the S&P monitoring dashboard, refine the security policy accordingly, and launching a new DevOps loop where GeneSIS would deploy the new security policy in the external service or within the KPs as needed. In case external security policy checkers are used, it is advisable that they are deployed by GeneSIS too, when possible, so as to save time during deployment and redeployment loops.

4 Conclusion

In this deliverable the final validation and verification results of the project, including both the work carried out in the first and the second period, were presented. The ENACT tools were validated in each of the three use cases and specific validation and verification tests were performed for each of the DevOps scenarios. In addition, an overall evaluation of the performance of each enabler in the project, conclusions and recommendation for future development was made.

From the ITS use case perspective, the Use Case itself has been enriched and some of the challenges that the IoT systems bring to the rail environment could be solved. The adaptation of the rail scenario brought to the project has enabled quick deployment and monitoring capabilities of different trustworthy indicators. These factors ensure the IoT technology as a feasible solution into an environment with a remarkable charge of safety and security regulations. The addition of DevOps tools has solved the rail environment main challenge as it is ensuring that the ITS Gateways has a proper capacity and how to scale them. Moreover, the security in the On Board layer can be monitored and adapter to the IoT protocols and the new wireless network infrastructure can be tracked and its performance issues detected.

From the Digital Health use case perspective, the use case itself has evolved from being a prototype development in the beginning of the project to being a production ready system recently sold to customers and already deployed for remote supervision of several hundred patients suffering from chronical diseases such as COPD, Kidney and Diabetes. Moreover, it has been applied for remote supervision and following up of Corona patients in several municipalities in Norway. The support and exploration we have been able to do in ENACT in particular related to efficient DevOps processes, Security and trustworthiness aspects and exploring the ENACT enablers and have been significant to reach this success. In particular, the Personal Health Gateway, which is a core component managing the IoT and edge part of our service, the ENACT project has been significant for its evolution. Furthermore, the outcome of the ENACT project will be further explored to ensure a scalable and trustworthy service as we see both the potential and the need to further improve our DevOps process to cope with the scale of distribution we foresee. Thus, we have already initiated some following up projects and activities to further evolve on results from the ENACT project. In particular the deployment and orchestration enabler bundle where we have successfully applied for a following up R&D project that is receiving funding from the Norwegian Research Council
This Deliverable has presented evaluations and validations performed applying the Digital Health use case in ENACT. In particular the Orchestration and Continuous Deployment bundle, the context aware access control, the risk driven decision support enabler and the test and simulation enabler.

From the Smart Building use case perspective, the use case has enabled the SMOOL middleware to be used in both in the Smart Home and the Kubik facility scenario for the interoperation between the subnetworks, wired and wireless, using different protocols within the building space. The scenarios of the use case have benefited from the GeneSIS tool to carry out a secure deployment of the IoT applications over Edge and Cloud infrastructure even on devices not directly connected to Internet. The use case has validated several ENACT tools for secure and trustworthy actuation. In the Smart Building DevOps scenarios security policies has been injecting in the SMOOL middleware to develop secure actuation; the actuation conflicts between different IoT applications when trying to access de same actuator device or controlling the same physical variable has also been an important case study; and the behavioural drift between real and expected actuation has been successfully evaluated in a real environment. Finally, the Kubik facility scenario has validated the security and privacy monitoring and control tool by using network protocol analyser, signature-based intrusion detection and machine learning-based anomaly detection.

5 References

[1] D1.1 Use case definition and requirements, validation and evaluation

[2] D1.2 Case studies implementation first version

[3] D1.3 First evaluation and validation report

[4] D1.4 Case studies implementation final version

[5] Indra Sistemas S.A, Security in Rail IoT Systems: An IoT Solution for New Rail Services

[6] ENACT_Rail_Business_Apps_Interfaces_EDI_BOSC

[7] ENACT Canonical Data Model Specification

[8] See https://gitlab.com/enact/its-app/container_registry/1428142

Appendix A Evaluation and validation test materials

In this annex, we summarize all the materials we created during the evaluation and validation tests. A material can be a video of the test scenario, a repository containing the code and document to describe the tests, or the combination of both. Each material may correspond to multiple test cases and evaluates multiple KPIs (test cases and KPIs are defined in Section 3), and involve one or more enablers. It is worth noting that the links to SharePoint refers to confidential materials and are available on request. Other materials are publicly available.

A.1 ITS DOMAIN (RAIL)

A.1.1 Train Integration

The Use Case tests were defined in the D1.1 and confirmed in the D1.3. The metadata functionality was identified after the provision of the D1.1 during the D1.3 definition; therefore, there is not a tests defined for it besides the D1.1 tests. The test related with the metadata functionality is explained in the section 2.1.3.

The inauguration process, integrity process, hard reset, logistic and maintenance process were validated in the first period of the project and the results shown in the D1.3.

The test defined during the first period of the project for the D1.1 has been evolved during the project as the DevOps functionalities and the integration plan with the ITS Use Case are adapted during the project development due to different constrains. The next subsections are intended to explain the different adaptations that has been made on the D1.1 defined tests to explain the results stated in the section 2.1.

Regarding the T&S, the initial focus has been kept and, based on recorded behaviour of the Train operation, the tool in this area is able to provide hints about the scalability factor of the Use Case, the D1.1 and D1.3 focus is kept. Further details in the section 2.1.1.2.

Regarding the T&S, it is been observed that the ACM tool has not a direct interrelation in this Use Case as they are focus ion different aspects of the Use Case. The RCA is focus on the behaviours of the communication infrastructure independently of the operation status, but the ACM in the operation. Therefore, the initial tests scheme has been adapted to accomplish the same KPIs as it is stated in the section 2.1.1.3 and 2.1.1.5.

The videos that present the RCA, T&S, and ITS functionalities are available on demand in the following links:

- **ITS Use Case:**
  https://sintef.sharepoint.com/sites/enact/Delte%20dokumenter/Forms/AllItems.aspx?viewid=f9d9310c%2D941b%2D4514%2D8345%2Da99f8a94511e&id=%2Fsites%2Fenact%2FDelte%20dokumenter%2FWP1%20Case%20Studies%20and%20Validation%2FITS%20%2D%20DevOps%20tools%20videos%2FITS%20Use%20Case
- **RCA:**
  https://sintef.sharepoint.com/sites/enact/Delte%20dokumenter/Forms/AllItems.aspx?viewid=f9d9310c%2D941b%2D4514%2D8345%2Da99f8a94511e&id=%2Fsites%2Fenact%2FDelte%20dokumenter%2FWP1%20Case%20Studies%20and%20Validation%2FITS%20%2D%20DevOps%20tools%20videos%2FRCA
- **T&S:**
  https://sintef.sharepoint.com/sites/enact/Delte%20dokumenter/Forms/AllItems.aspx?viewid=f9d9310c%2D941b%2D4514%2D8345%2Da99f8a94511e&id=%2Fsites%2Fenact%2FDelte

%20dokumenter%2FWP1%20Case%20Studies%20and%20Validation%2FITS%20%2D%20DevOps%20tools%20videos%2FT%26S

A.1.2 Actuation conflict management and Behavioural Drift Analysis for trains

The initial approach for this section was monitoring the On Track data besides the On Board data that could be deployed by the GeneSIS tool in an infrastructure used validated with the T&S tool and evaluated with the RCA tool.

The T&S and RCA part are performed as expected. However, the GeneSIS tool role is modified as the business train data is not deployed. The GeneSIS role is enabling to the BDA tool to monitor the data, but the functionality that reports the business data is completely managed by the ITS manager and not deployed by the GeneSIS tool. Therefore, the initial tests scheme has been adapted to accomplish the same KPIs as it is stated in the section 2.1.1.1 and 2.1.1.4.

The video that presents the ACM and BDA functionalities, can be checked in the following link into the ENACT project YouTube channel: https://youtu.be/RXHkpAyxmt0 and https://youtu.be/mokA22r20a4

A.1.3 Trustworthy deployment of software on train gateways

It must be remarked that the tests described in the D1.1 and D1.3 for the software update were focus on rail data deployment; however, in the middle of the project was shown that the GeneSIS tool could be exploited to update not rail data but configurations into the devices that interacts with its security capabilities.

This new focus is compatible as the other ENACT tools models can be deployed in the ITS infrastructure and the configuration enable or disable the operation and the integration of the components for the ITS Use Case. This view is further explained in the section 2.1.1.1.

The video that presents the GeneSIS functionalities, can be checked in the following link into the ENACT project YouTube channel: https://youtu.be/BLmbagBtLo0

A.1.4 Security monitoring of train gateways

The initial focus has been kept and, based on learned behaviour of the Train operation, the tools in this area are able to interact with the security aspects of the ITS Use Case, the D1.1 and D1.3 focus is kept. Further details in the section 2.1.1.6.

The videos that present the S&P functionalities working in the ITS use case are uploaded in the ENACT project intranet, and they are not public since they show real traffic data and incidents. However, the videos are available upon request, under the sharing conditions of Tecnalia.

A.2 DIGITAL HEALTH

A.2.1 Dynamic introduction of Context-Aware Access Control into the gateway fleet

For CAAC:

- Video presenting the functioning of the CAAC infrastructure: https://www.youtube.com/watch?v=JKAYQqx1ShU
- Testing platform that provides CAAC in SaaS mode: https://enact-caac-my.evidian.com
- The CAAC authorization server is available at: https://enact-caac-auth.evidian.com
- The User's Token Management UI is available at: https://enact-caac-my.evidian.com/oidc-provider
- EVIDIAN Proprietary software. The user's guide is available on demand.
- Mock-up application provided to be able to tune the taking into account of the contextual events by the risk server: https://enact-caac-riskserver-ui.evidian.com/

For the deployment of CAAC code via DivEnact and GeneSIS:

- Video presenting the fleet deployment and the CAAC code into Tellu gateway and Arduino board is available online: https://www.youtube.com/watch?v=a22YP4TB-To
- DivEanct Deployment model used for this scenario is available at https://gitlab.com/enact/divenact/-/raw/master/sample-data/templates-03-10.json
- GeneSIS model for this scenario is available at: https://gitlab.com/enact/divenact-agent/-/blob/master/modules/GenesisBroker/model_test.json

A.2.2 Risk management of Tellu DevOps process and the evidence collection for regulation-compliance

Simulation and Tellu gateways

A.2.3 Test and Simulation enabler – simulate TellU device

- Video demo: https://youtu.be/tTaL9quBG0o
- Test and Simulation enabler source code: https://gitlab.com/enact/test_and_simulation
- eHealth Gateway source code: https://github.com/TelluIoT/gateway-enact

A.3 SMART BUILDING

A.3.1 The Smart Home tests

- All the models and code used for running the overall scenario is available on Gitlab in a dedicated repository: https://gitlab.com/enact/smarthome-demo
  - GeneSIS and ThingML models can be found at: https://gitlab.com/enact/smarthome-demo/-/tree/master/genesis
  - S&P SMOOL client code: https://gitlab.com/enact/smarthome-demo/-/tree/master/security
  - ACM models and conflict manager code: https://gitlab.com/enact/smarthome-demo/-/tree/master/acm
  - T&S models and data records: https://gitlab.com/enact/smarthome-demo/-/tree/master/TaS
  - Application code: https://gitlab.com/enact/smarthome-demo/-/tree/master/smarthome
- Descriptive video of the Smart Home tests: https://sintef.sharepoint.com/:v:/r/sites/enact/Delte%20dokumenter/WP1%20Case%20Studies%20and%20Validation/Videos/ENACT%20Recording%20of%20the%20video-20210323_141233-Meeting%20Recording%20(1).mp4?csf=1&web=1&e=zoxrL0
- Descriptive video of the Security Control and GeneSIS integration: https://www.youtube.com/watch?v=9Y0e5NIQFO4

A.3.2 SMOOL generic KPs for Enact

- Base Producer and Consumer with built-in security https://gitlab.com/enact/smool_enact/
- SMOOL official repository with server, updatesite, and documentation https://bitbucket.org/jasonjxm/smool
- Videos on developing SMOOL KPs: https://youtube.com/playlist?list=PLuc0lPfwaR5EQBHdUqR2AUXEYf4HTrPlr
- Video showing integration between SMOOL and THINGML: https://youtu.be/u-2u2vwxFvY

A.3.3 Security monitoring for Kubik

The videos that present the S&P monitoring functionalities working in the Kubik infrastructure are uploaded in the ENACT project intranet, and they are not public since they show real traffic data and incidents. However, the videos are available upon request, under the sharing conditions of Tecnalia.

A.3.4 Security control for Kubik

- Video of S&P Control enabler functionality in Kubik: https://youtu.be/mfT_AwfkXNcSecurity KP for Enact: https://gitlab.com/enact/smool_enact/-/tree/master/ENACTSecurity

A.3.5 Lighting comfort control: Actuation conflict monitoring tests on Kubik

- Descriptive video of the Kubik infrastructure and the Supervisory control and data acquisition (SCADA): https://gitlab.com/enact/kubik_enact/-/blob/master/videos/KUBIK_Smart_Building_Scada.mp4
- Node-RED code for lighting and blind control on the ground floor of Kubik. This code presents an actuation conflict when there is enough light outside and the lights are on, which is energy inefficient and useless.

https://gitlab.com/enact/kubik_enact/-/blob/master/Node-RED_code/KUBIK_blinder_light_example.json

- Node-RED code for lighting and blind control on the ground floor of Kubik corrected by ACM enabler:
  https://gitlab.com/enact/kubik_enact/-/blob/master/Node-RED_code/KUBIK_blinder_light_example_ACM_corrected.json
- Video showing how ACM enabler is used to correct the conflict between lighting control and blind control:
  https://gitlab.com/enact/kubik_enact/-/blob/master/videos/KUBIK_Lighting_Control_using_ACM_tool.mp4

A.3.6 User comfort control via online learning

- Video showing how OLE is used: https://www.youtube.com/watch?v=-0OW3Jts15s
- CSV files containing data recorded during evaluation experiments (TO2.3 & Test 3.5.0.1): https://gitlab.com/enact/online-learning-enabler/-/tree/supplementary_material/experiment_data
  - Further details concerning evaluation can be found in D3.3
- Deployment with GeneSIS (Test 3.5.0.2): https://gitlab.com/enact/online-learning-enabler/-/tree/supplementary_material/deployment_with_GeneSIS
  - Video shows:
    - Loading of deployment model
    - Deployment
    - OLE & simulation is started automatically
    - OLE is accessed
- OLE/BDA integration (TO2.4) (https://gitlab.com/enact/online-learning-enabler/-/tree/supplementary_material/integration_with_BDA):
  - OLE and BDA was deployed with docker and exchanges data via MQTT
  - Video shows how integration is set up
  - Further details in D3.3