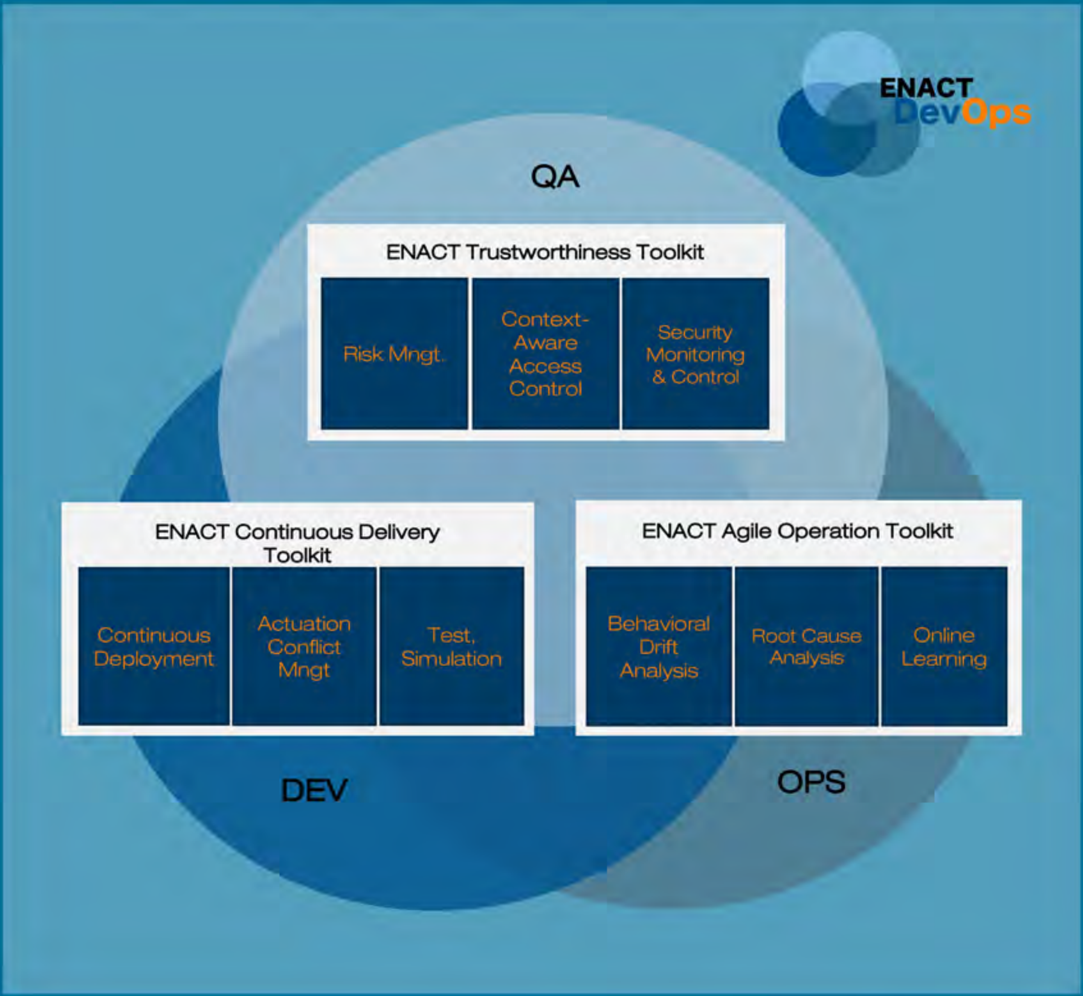
nowopen in Technology

DevOps for Trustworthy Smart IoT Systems

Nicolas Ferry, Hui Song, Andreas Metzger and Erkuden Rios (Editors)


ENACT DevOps
QA
ENACT Trustworthiness Toolkit
Risk Mngt.
Context-Aware Access Control
Security Monitoring & Control
ENACT Continuous Delivery Toolkit
Continuous Deployment
Actuation Conflict Mngt
Test, Simulation
ENACT Agile Operation Toolkit
Behavioral Drift Analysis
Root Cause Analysis
Online Learning
DEV
OPS

now

DevOps for Trustworthy Smart IoT Systems

Nicolas Ferry, Hui Song
Andreas Metzger
and Erkuden Rios
(Editors)

now
the essence of knowledge

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

ISBN: 978-1-68083-824-4
E-ISBN: 978-1-68083-825-1
DOI: 10.1561/9781680838251

Suggested citation: Nicolas Ferry, Hui Song, Andreas Metzger and Erkuden Rios (eds.). (2021). *DevOps for Trustworthy Smart IoT Systems*. Boston–Delft: Now Publishers

Funded by the European
Union (EU) as part of the
Horizon 2020 program

Table of Contents

Foreword

Explosive growth in devices and software connected to the internet has led to current estimates of 46 billion “things” connecting and integrating on the internet. Increasing numbers of applications are being facilitated by the use of technology attached to the internet as part of this Internet of Things (IoT). The scale of IoT has inevitably led to the increasing complexity of any solution. Theoretical solutions must quickly give way to pragmatic technology. Smart IoT Systems (SIS) have a highly distributed infrastructure that relies on a wide range of topologies including, but not restricted to, cloud computing, edge computing, and closed or open networks. It requires tools and technology that are reliable, robust, adaptable and secure. Improvements in software, methodology, AI, data-analysis and decision-making are leading to practical applications that will improve the target systems in which they are implemented. The ENACT project indicates that there are many feature and functionality gaps in both the applications and enablers present in this environment and aims to close some of the significant gaps.

The ENACT project has been funded by the European Commission under its H2020 program. The project consortium consists of twelve member organisations spread across the EU as a whole. The funding has allowed the research and development of three toolkits covering trustworthiness, continuous development and agile operation. The emphasis of the toolkits is to build applications quickly and maintain those applications as the application target’s circumstances change. The development of the toolkits has benefitted from the research excellence that has pervaded all stages of the project.

Proof of concept use cases will support the development and testing of the toolkits. Use cases that are significant in scope and demonstrate the challenges to be met by the project researchers have been chosen and are discussed in this book. The three case study domains are in the domains of eHealth, Smart buildings and Intelligent Transport Systems.

In the context of eHealth both wellness and preventative medicine programs are being seen as the best direction for the ever-growing healthcare sector. Instead of reactive medicine proactive medicine is now the preferred approach supported by the use of technology to move beyond health and disease monitoring to a continuum of support for the healthy to maintain their wellbeing. This is individualised by the use of machine learning and activation of AI systems capable of making diagnostic forecasts based on data collected.

In the post Covid 19 world smart buildings will become more pervasive, providing the facilities to monitor not just temperature and light but air quality, security and general health and wellbeing of the building users. Air quality monitors should be able to detect any airborne viruses, extract them or direct the airflow away from users. Individualised warnings or cautions should ensure that only people who are vulnerable or who are possibly threatened by poor air quality can be warned or relocated.

Intelligent transport systems are slowly being developed that take advantage of IoT devices to monitor and manage areas of the transport environment that were not previously managed. Autonomous cars use a variety of sensing and analytical tools to create a world view in which they can operate. Rail services also need to understand the context in which they move. Failed signals or unhitched rolling stock can pose an immediate danger on the track. The physical infrastructure of a rail system uses resources that take time to be implemented but the human machine interface of equipment/driver/passenger/bystander is adapting all the time. Prediction of component or train integrity failures will ensure a more secure and safe solution. Adaptable systems will enhance delivery and business models.

Each of these case studies needs tools for developing applications quickly, ensure that the optimum infrastructure can be built, managed and maintained. The whole environment must be able to adapt to changes rapidly and required updates or new applications can be delivered quickly and simply integrated into the whole infrastructure. The Dev-Ops philosophy is one of the foundations of the project and will support rapid and agile development and continuous delivery that can be demonstrated in these case studies.

I have been researching and writing about innovation and solutions development for many years and am familiar with the pragmatic research and demonstrations that are part of the European Commission framework programs. I attended the ENACT project kick-off meeting and later as a member of the project advisory board attended project reviews. The project case studies are in domains that I am familiar with from my work on automation and collaborative robotics. The domain specific problems in the use cases can be resolved with the general toolkits developed by the project and will aid decision making, development, deployment and maintenance. This book contains detailed and well written chapters that cover the most important areas of the project. The progress made in the project is excellent and I have been able to observe their progress over the project's duration. In my experience it is always difficult to manage such a large project, but the consortium has managed this well. This book has many interesting topics but I found that the book chapter on "Looking Ahead" was most interesting as it highlights future issues with integration and customisation of applications, an area that has been at the heart of my own work over many years. The authors clear view of what still remains to be done and the potential for further research is insightful.

Peter Matthews
31st March 2021

Glossary

A

ACM - *Actuation Conflict Management*. 9, 10, 18, 97, 98, 100–103, 111–114, 116–119, 226, 229, 238, 253–255

AGG - *Attributed Graph Grammar*. 97

AI - *Artificial Intelligence*. 152, 154, 170

API - *Application Programming Interface*. 69, 79, 83, 89, 146, 164, 165, 170, 194, 195, 250, 260

AVD - *Automatic Vulnerability Detector*. 26, 49

B

BDA - *Behavioral Drift Analysis*. 11, 18, 119, 137, 226, 229, 238–240, 256

C

C-ITS - *Cooperative Intelligent Transportation System*. 47, 49, 50, 52

CAAC - *Context-Aware Access Control*. 12, 162, 166

CAPEC - *Common Attack Pattern Enumeration and Classification*. 26, 44, 47, 49, 52

CMW - *Communication MiddleWare*. 230–233

CPS - *Cyber Physical System*. 16

CWE - *Common Weakness Enumeration*. 26, 44, 46, 47, 49, 52

D

DBN - *Dynamic Bayesian Networks*. 105

DevOps - *Development and Operation*. 2, 3, 6–20, 59, 60, 64, 67, 77–81, 83, 85, 86, 89, 95–99, 101, 103, 104, 109, 110, 118, 119, 170, 175, 188, 193, 195, 196, 198, 218, 221–223, 225–227, 229, 233–235, 237, 239, 240, 242, 259

DEVS - *Discrete EVent system Specification formalism*. 97

DevSecOps - *Development, Security, and Operation*. 59, 60, 78, 81, 84–86

DFD - *Data Flow Diagram*. 25, 26, 44–46

DMI - *Driver Machine Interface*. 232

DPIA - *Data Protection Impact Assessment*. 28, 29

DPO - *Data Protection Officer*. 31

DS - *Data Subject*. 24–43, 46, 47, 49, 50, 52, 53, 55

E

ECA - *Electronic Clearing Service*. 117

F

FSM - *Finite State Machines*. 101

G

GDPR - *General Data Protection Regulation*. 9, 20, 24–30, 36–44, 46, 49, 53–56

GNSS - *Global Navigation Satellite System*. 230, 231

H

HVAC - *Heating, Ventilation and Air-Conditioning*. 18, 123, 129, 130, 133, 136, 138, 252, 254

I

IaC - *Infrastructure as Code*. 60, 63

IAM - *Identity and Access Management*. 162

ICT - *Information and Communication Technology*. 64, 215, 216, 262

ID - *Identifier*. 49–54, 208

IDE - *Integrated Development Environment*. 79, 253

IEC - *International Electrotechnical Commission*. 1, 152

IOHMM - *Input/output Hidden Markov Model*. 105, 106, 108, 138

IOI - *Items Of Interest*. 38, 40, 42

IoT - *Internet of Things*. 1–4, 6–14, 16–18, 20, 23–25, 47, 54–56, 59–70, 78–84, 86, 89, 94–98, 102, 104, 110, 118, 123, 138, 139, 142–145, 147–162, 164, 165, 170, 171, 173–181, 183–185, 187, 188, 190, 192, 193, 199, 201, 207–209, 211, 212, 218, 221–223, 226, 228, 234–236, 240–244, 247–249, 252–257, 259, 264

ISO - *Information Systems and Organizations*. 152

IT - *Information Technology*. 52, 62, 161, 162, 170, 218

ITS - *Intelligent Transportation System*. 4, 17, 94, 95, 195, 199, 204, 232–237, 239

K

KDT - *Key Digital Technologies*. 259–264

KP - *Knowledge Processor*. 144–151, 157, 158, 160

L

LINDDUN - *Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of Information, Unawareness, and Non-compliance*. 24, 25, 27, 36–38, 42, 44–46, 49, 51, 56

M

MAC - *Media Access Control*. 235

MDE - *Model-Driven Engineering*. 61, 101, 117

MDP - *Markov Decision Process*. 128–130, 137

MFCC - *Mel-Frequency Cepstral Coefficient*. 112, 113

Mon&Con - *Monitoring and Control*. 152, 226, 232, 234

MQTT - *Message Queuing Telemetry Transport*. 110, 154, 155, 181, 183, 184, 188, 199, 204, 222, 248–250

MUSA - *MUlti-cloud Secure Applications*. 24

MVC - *Model – View – Controller*. 68

N

NAS - *Network Attached Storage*. 110

NIST - *National Institute of Standards and Technology*. 15, 31, 151

O

OLE - *Online Learning Enabler*. 11, 126, 257

OWASP - *Open Web Application Security Project*. 24, 25, 27–33, 49, 52–54

P

PDP - *Policy Decision Point*. 163

PDP4E - *Privacy and Data Protection Engineering*. 24, 44, 46, 49, 56

PEP - *Policy Enforcement Point*. 163

PHG - *Personal Health Gateway*. 19, 20, 221, 223

PII - *Personally Identifiable Information*. 16

R

RCA - *Route Cause Analysis*. 12, 13, 96, 175, 198–212, 227, 239, 240

REST - *Representational State Transfer*. 164, 194, 250

RFID - *Radio Frequency Identification*. 230–232

RL - *Reinforcement Learning*. 123–138

S

S&P - *Security and Privacy*. 152, 226, 232–235, 240

SCA - *Software Communications Architecture*. 62, 63

SDR - *Software-Defined Radio*. 62

SIB - *Semantic Information Broker*. 144

SIS - *Smart IoT System*. 1–4, 6–12, 14–20, 23, 24, 47, 59–61, 65–67, 70, 71, 78–81, 89, 94–97, 99, 101–108, 110, 112–114, 118, 119, 123, 124, 129, 138, 142, 143, 148, 149, 151–156, 159–161, 170, 171, 177–180, 183, 184, 190, 196, 198, 199

SLR - *Systematic Literature Review*. 63

SMT - *Satisfactory Modulo Theories*. 64, 65

SSAP - *Source Service Access Point*. 144, 145

T

TaS - *Test and Simulation*. 10, 18, 174–181, 183–185, 187–197, 211, 212

TBM - *Transferable Belief Model*. 105

TMS - *Train Management Systems*. 226

U

UML - *Unified Modelling Language*. 44

V

VPN - *Virtual Private Network*. 51

W

WAM - *Web Access Manager*. 162, 167–170

WIMAC - *Workflow and Interaction Model for Actuation Conflict management*. 97

WSAN - *Wireless Sensor Actuator Network*. 229

WSN - *Wireless Sensor Network*. 195, 226, 230–232

WTI - *Wireless Train Integrity*. 225, 227, 229

X

XACML - *eXtensible Access Control Markup Language*. 162–164

Z

ZCR - *Zero Crossing Rate*. 113

DOI: 10.1561/9781680838251.ch1

Chapter 1

Introduction

By Erkuden Rios, Nicolas Ferry, Hui Song and Andreas Metzger

Internet of Things (IoT) systems are evolving towards what we denote as Smart IoT Systems (SIS) – *i.e.*, systems involving not only sensors but also actuators with control loops distributed all across the IoT, Edge and Cloud infrastructure.

However, the capacity in building novel and innovative SIS faces specific challenges, that entail (i) how to efficiently build and operate new value-added software across IoT, edge and cloud infrastructures, (ii) how to close the loop of sensing and actuation, and (iii) how to establish trustworthiness in these systems. Whilst point (iv) is already critical in classical IoT systems: according to the IEC report on smart and secure IoT platforms [5], security, trust, privacy and identity management are major challenges in today's IoT systems, this is all the more exacerbated when actuators are involved.

One of the fundamental research questions concerning these issues is: "how can we tame the complexity of developing and operating smart IoT systems, which

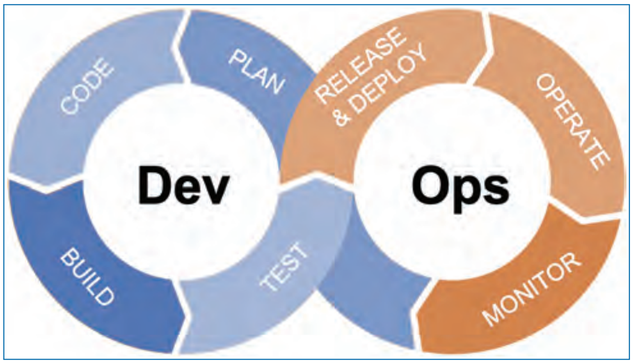


Figure 1.1. The generic DevOps life-cycle model.

(i) consist of software running on all types of resources along the IoT-edge-cloud continuum, (ii) involve sensors and actuators and (iii) need to be trustworthy?". The answer to these questions has formed the core of the Horizon 2020 project ENACT [1, 3]. The overall ambition of ENACT was to expand current DevOps methods and solutions to support the development and operation of trustworthy Smart IoT Systems.

DevOps has established itself as a software development life-cycle model that encourages developers to continuously patch, update, or bring new features to the system under operation without sacrificing quality [8]. By enabling DevOps in the realm of SIS, ENACT not only facilitates the development and operation of SIS but also enables the continuous and agile evolution of SIS, which is necessary to adapt the system to changes in its environment, including such as newly appearing trustworthiness threats. ENACT supports DevOps practices during the development and operation of trustworthy smart IoT systems by offering software tools, called "enablers", for each of the seven stages of the DevOps life-cycle model as depicted in Figure 1.1.

- **Plan:** ENACT supports privacy and security risk assessment enabling the risk-driven planning of IoT systems development cycles as well as the smooth transition towards the code stage.
- **Code:** First, ENACT evolves recent advances of the ThingML language and generators to support modelling of system behaviours and generation of code executable across the whole IoT, edge and cloud continuum. Second, ENACT provides a model-based solution to automatically identify and solve conflicts when multiple applications manage actuators.
- **Test:** Targeting the constraints related to the distribution and infrastructure of IoT systems, ENACT enables continuous testing of SIS in an environment by emulating and simulating IoT and Edge infrastructures.

- **Release and Deploy:** ENACT provides novel deployment modelling languages and the corresponding execution engines to support the continuous and automatic fleet deployment, by assigning multiple deployments to many devices in the fleet, without human interaction. It enables deployment from the IoT to the cloud ends with security as a first-class concern.
- **Operate:** ENACT provides enablers for the automatic adaptation of IoT systems based on their run-time context, including smart preventive security mechanisms such as access control. In addition, ENACT offers machine learning capabilities at runtime in order to deliver self-adaptive SIS. Such automatic self-adaptation addresses the issue that the management complexity of open-context IoT systems exceeds the capacity of human operation teams, and by this, improve the trustworthiness of the smart IoT system execution.
- **Monitor:** ENACT has delivered innovative mechanisms to observe and analysis (i) the status of a SIS including security and privacy aspects at all the network, system, and application levels, (ii) failures, (iii) the overall effectiveness of the SIS in reaching its goals.

ENACT was part of a cluster of related H2020 projects all contributing to IoT security [2]. Among the eight projects that formed the cluster, two are most notably related to ENACT and share common objectives. The Semiotics project[1] also considers SIS with a specific focus on the management of actuators. The project proposes a pattern-driven framework, built upon existing IoT platforms, to enable and guarantee secure and dependable actuation and semi-autonomic behaviour in IoT applications. While not specifically focusing on DevOps, one of the technical objectives of the Brain-IoT project[2] was to facilitate the rapid model-based development, integration, and deployment of interoperable IoT solutions that support smart cooperative behaviour involving actuation in IoT scenarios.

This book describes the ENACT project outcomes (cf. Figure 1.2) and how they solve major challenges in the DevOps of trustworthy SIS. The overall approach pursued in the ENACT project is introduced in Chapter 2, and the chapters following afterwards detail the outcomes of ENACT. In Chapter 3 the privacy and security risk assessment and management in SIS is discussed, and the ENACT enabler dealing with risks is presented. Chapter 4 is focused on deployment support offered by ENACT and the deployment and diversification methods and enablers are detailed therein. Chapter 5 deals with the issues of actuation conflict resolution in SIS that

1. https://www.semiotics-project.eu
2. http://www.brain-iot.eu

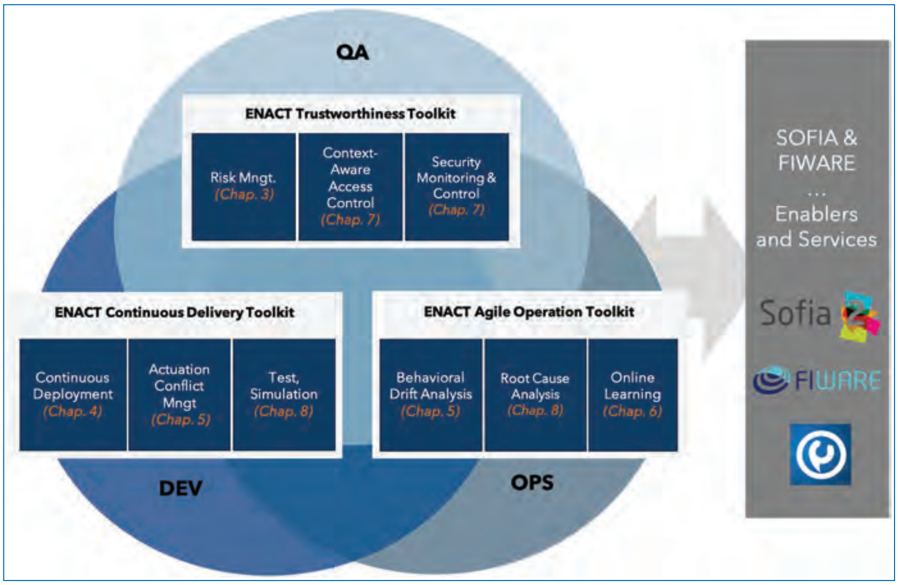

QA
ENACT Trustworthiness Toolkit
Risk Mngt. (Chap. 3)
Context-Aware Access Control (Chap. 7)
Security Monitoring & Control (Chap. 7)
ENACT Continuous Delivery Toolkit
Continuous Deployment (Chap. 4)
Actuation Conflict Mngt (Chap. 5)
Test, Simulation (Chap. 8)
ENACT Agile Operation Toolkit
Behavioral Drift Analysis (Chap. 5)
Root Cause Analysis (Chap. 8)
Online Learning (Chap. 6)
DEV
OPS
SOFIA & FIWARE
...
Enablers and Services
Sofia
FIWARE

Figure 1.2. ENACT project results.

include actuators, and how to detect and analyse behaviour deviations at SIS operation from those designed when building the SIS. In Chapter 6 reinforcement learning techniques are studied as the ENACT approach for continuously ensuring and improving the quality of SIS during operations. Chapter 7 explains all the details of the ENACT support to security aspects of SIS, including context-aware access control enabler, security monitoring enabler, as well as security control through capabilities embedded in IoT platforms. Chapter 8 describes the ENACT enablers dedicated to the SIS verification and validation activities, including the support to testing, simulation and root cause analysis. Chapters 9, 10 and 11 explain the real IoT system use cases where the ENACT enablers were validated, dedicated to eHealth, Intelligent Transport Systems (ITS) and Smart Buildings domains, respectively. Chapter 12 concludes the book with an outlook on future research challenges and opportunities.

References

[1] ENACT Consortium. *ENACT: Development, Operation, and Quality Assurance of Trustworthy Smart IoT Systems*, 2018. URL: https://cordis.europa.eu/project/id/780351.

[2] Enrico Ferrera *et al.* “IoT European security and privacy projects: Integration, architectures and interoperability. In: *Next Generation Internet of Things. Distributed Intelligence at the Edge and Human Machine-to-Machine Cooperation* (2018).

[3] Nicolas Ferry *et al.* “ENACT: Development, operation, and quality assurance of trustworthy Smart IoT Systems”. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment.* Springer. 2018, pp. 112–127.

[4] Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 2010. ISBN: 860-1401501176.

[5] IEC. *IEC White Paper: IoT 2020: Smart and secure IoT platform*. IEC report, 2019. URL: https://basecamp.iec.ch/download/iec-white-paper-iot-2020-smart-and-secure-iot-platform/.

DOI: 10.1561/9781680838251.ch2

Chapter 2

The ENACT Approach

By Nicolas Ferry, Hui Song, Erkuden Rios and Andreas Metzger

Smart IoT Systems (SIS) are the next generation of IoT systems that span across the complete computing continuum, from IoT via Edge/Fog to the Cloud, with local data analytics, decision making, and actuators involved. Software plays a key role in such systems. The systems' increased complexity, the unpredictability of their environment, as well as the changes in their requirements and infrastructure are many factors that can result in new threats hindering their trustworthiness. The proper functioning and correctness of such systems is critical especially when they control actuators that can have a direct impact on the physical world. The ability of these systems to continuously evolve and adapt to these changes is decisive to ensure and increase their trustworthiness, quality and user experience. Currently, DevOps is the mainstream practice in the software and Cloud industry to foster continuous evolution of software systems. DevOps promotes a rapid and efficient value delivery to the market, through a tight collaboration between the developers and the teams that deploy and operate the software systems. DevOps seeks to decrease the gap between product design and its operation by introducing software design and development practices and approaches to the operation domain and vice versa [8].

When the ENACT project was created in 2017, there was no DevOps support for trustworthy Smart IoT Systems [12, 34]. Even if DevOps is not bound to any application domain, many challenges appear when the IoT intersects with DevOps, in particular, due to the lack of key enabling tools. ENACT focused specifically on the following three challenges [3].

The first key challenge, as opposed to Cloud environments which are relatively reliable and homogeneous, is the wide diversity that characterizes SIS, not only in terms of hardware but also in terms of their software stack. There is typically a lack of coherent languages, abstractions, security and privacy solutions that can be used to support development and the orchestration of software and their deployment across heterogeneous devices.

Second, SIS are by nature massively distributed on top of a highly heterogeneous and geographically-distributed infrastructure, which means that software is more complex to apprehend, develop, operate, and maintain than on top of Cloud infrastructures. Each device has a unique operational context, in terms of hardware capacity, end-user preference, exposure to security risks, role in the whole data flow, connection to sensors and actuators, etc. This context is dynamic and often unpredictable, *e.g.*, the volume of data may change, the network connectivity among devices can be unstable. Therefore, the management and operation of each software module, *e.g.*, where to place it, when to deploy it, how to configure it, and how to monitor it, etc, needs to be handled individually and continuously to fit its unique and evolving context. For large scale SIS which can include thousands of devices, handling each device individually inevitably leads to enormous operational effort and cost, which, as identified by Gartner, "can easily exceed the project's financial benefits."[1]

Third, SIS can have an impact on the physical world through actuators. There is a need to properly manage these actuators and to ensure that such systems and in particular the software deployed on these systems always work within safe operational boundaries. Only a few approaches exist in the literature focusing on the management of actuation conflicts, and none are meant to be used in a DevOps context.

1. https://www.gartner.com/en/newsroom/press-releases/2018-12-03-gartner-says-the-future-of-it-infrastructure-is-always-on-always-available-everywhere

These key challenges had to be addressed to enable DevOps for trustworthy Smart IoT Systems. In this chapter we present how the overall approach followed in the ENACT project proposes to evolve existing DevOps methods and techniques to support the development and operation of Smart IoT Systems, which (i) are distributed, (ii) involve sensors and actuators and (iii) need to be trustworthy (*i.e.*, trustworthiness refers to the preservation of security, privacy, reliability, resilience, and safety [13]).

The remainder of the chapter is organized as follows. Section 2.1 introduces the overall ENACT approach and lists the enablers that will form the core contribution of ENACT supporting the DevOps of SIS. Section 2.1.4 details how these enablers can be organized together to form a comprehensive and continuous DevOps Framework. Section 2.2 summarizes how the developed solutions facilitate the development and operation of SIS that are trustworthy. Finally, Section 2.3 reports on the three use cases of the project and how they supported validation of the ENACT enablers.

2.1 ENACT Enablers to Deliver DevOps for SIS

To foster the adoption of DevOps practices in the realm of SIS, ENACT's approach is to deliver a set of enablers (*i.e.*, tools and services) that support the continuous development, evolution and operation of SIS. These enablers are designed to integrate with DevOps, Cloud, and Edge services and are loosely coupled, providing SIS providers with the ability to pick the enablers that best fit their needs. In other words, it is not necessary to gather the whole ENACT framework to benefit from one or more of these enablers, and these can be integrated as part of existing DevOps pipelines. As depicted in Figure 2.1, the different enablers contribute to different stages within the DevOps life-cycle and, overall, the ENACT Framework contributes to all the DevOps stages. In the following we provide an overview for each of the enablers, which are detailed in the remaining chapters of this book.

2.1.1 Enablers for the Development Phase

The following three enablers provide specific support for the development of trustworthy SIS.

Risk Management enabler: This enabler provides concepts and tools for the agile, context-aware, and risk-driven decision support and mechanisms for application developers and operators to support the continuous delivery of trustworthy SIS [12]. By leveraging the evidences collectors provided by the enabler,

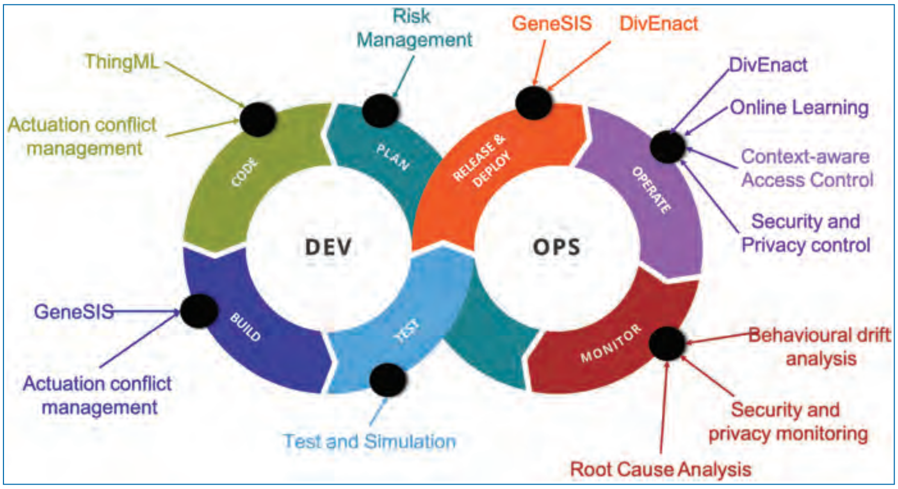

Risk Management
GeneSIS
DivEnact
ThingML
Actuation conflict management
DivEnact
Online Learning
Context-aware Access Control
Security and Privacy control
CODE
PLAN
RELEASE & DEPLOY
OPERATE
DEV
OPS
BUILD
TEST
MONITOR
GeneSIS
Actuation conflict management
Test and Simulation
Behavioural drift analysis
Security and privacy monitoring
Root Cause Analysis

Figure 2.1. Contribution of the ENACT tools to the DevOps lifecycle.

organizations use it not only to assess risks but also to monitor and control treatment implementation and effectiveness during the development and operation of SIS, enabling the treatment of security and privacy risks together and making them actionable for software engineers. This makes this enabler the first DevOps-enabled continuous risk control solution, improving software development and operation organizations' awareness on risks. In addition, it facilitates compliance with standards such as ISO 27001 and regulations such as GDPR, in near real-time. Further details about this enabler can be found in Chapter 3.

ThingML: ThingML [4] is an open source IoT framework that includes a language and a set of generators to support the modelling of system behaviours and their automatic derivation across heterogeneous and distributed devices at the IoT and edge end. The ThingML code generation framework has been used to generate code in different languages, targeting around 10 different target platforms (ranging from tiny 8 bit microcontrollers to servers). A challenge for approaches such as ThingML is how to properly log, monitor and debug the generated programs. Indeed, to fully benefit from the approach, such logging should be performed by relating to the concepts of the original abstraction level. To address this challenge, ThingML has been extended with an automated,platform-independent and easy to use logging mechanism to ThingML developers. This logging approach aims at providing log information about the execution of their ThingML programs, in terms of ThingML concepts being executed.

Actuation Conflict Management enabler (ACM): Actuation conflicts can occur when concurrent applications have a shared access to an actuator and when actuators produce actions within a common physical and local environment, whose effects are contradictory. This enabler supports the identification and resolution of direct and indirect actuation conflicts as part of a DevOps pipeline in a platform independent and technology agnostic way [6]. DevOps team can integrate the ACM solution as part of their DevOps pipeline to detect automatically direct and indirect actuation conflicts in a complex SIS. Off-the-shelf actuation conflict managers are automatically injected into the SIS. New actuation managers can be designed using a tool-supported domain-specific modelling language and checked against logical and temporal properties. While traditionally, the management of actuation conflicts is handled at the code level, the ACM enabler applies over and abstract representation of the SIS that is decoupled from its detailed code enabling the detection, analysis and resolution of actuation conflicts as part of a typical DevOps process. Verification mechanisms ensure the conflict management solution injected into the SIS satisfies temporal and logical properties making DevOps teams confident to place it in the system. Further details about this enabler can be found in Chapter 5.

Test and Simulation enabler (TaS): Software testing is a crucial step of any software development process, especially in DevOps. Having access to a production-like environment that reproduces the same conditions where a piece of software would run is usually tricky or close to being an impossible task. This is exacerbated in IoT environments where (i) developers need to test their applications to ensure trustworthiness requirements, including scalability, are met, and (ii) building a large-scale testbed that includes a realistic physical infrastructure of devices and sensors can quickly be expensive. The test and simulation enabler provides a light-weight, user-friendly approach for simulating large number of IoT devices and cyber-attacks, in order to set up the testing environment and test SIS in a cost-effective way. The enabler goes beyond the state of the art on sensor and actuation simulation solutions that typically reproduce how the devices behave according to the physical environment. Instead, it focuses on the pure software simulation, by reproducing how devices interact with software in the IoT system.

2.1.2 Enablers for the Deployment Phase

Within the DevOps life-cycle, deployment is typically the activity that bridges development and operation activities. The following two enablers provide specific support for the deployment of trustworthy SIS.

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, Edge to Cloud [20]. 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 modeling elements thus promoting security-by-design. Further details about this enabler can be found in Chapter 4.

Fleet Management and Diversity enabler (DivEnact): This enabler, also known as DivEnact, supports automatic software deployment for IoT applications that comprise a large fleet of devices, and maintains software diversity among the fleet [44]. It provides DevOps teams with a mean to deploy a new software version into the abstract fleet, without worrying about what exact devices are in the fleet, their contexts, and whether they are online or not. DivEnact maintains the devices and their contexts in the fleet, the software variants, and assign the variants to the appropriate devices depending on their contexts. Further details about this enabler can be found in Chapter 4.

2.1.3 Enablers for Operation Phase

Finally, the following four enablers focus on supporting the operation and monitoring of SIS.

Behavioral Drift Analysis enabler (BDA): The complex nature of the cyber-physical environment in which a SIS operate makes it impossible for DevOps teams to predict if, once under operation, the system will behave as expected during development. For instance, many unanticipated surrounding physical processes may disrupt and hamper the SIS from achieving its goal. The Behavioral Drift Analysis enabler provides a novel way to overcome this issue by shifting the monitoring and analysis from the internal of the system to its context by observing and analysing the effects of the commands sent to the actuators on the cyber-physical context of the SIS [29]. This makes the approach generic an applicable to any SIS independently of its implementation and it makes it non-invasive in the sense that it does not require any modification of the applications. DevOps teams can use this enabler during operation as a monitoring solution to detect symptoms indicating that the effects of the system on its environment are no longer as expected and to understand this loss of effectiveness. Further details about this enabler can be found in Chapter 5.

Online Learning enabler (OLE): To develop a self-adaptive SIS, software engineers have to create self-adaptation logic encoding when the SIS should execute which adaptation actions. However, developing self-adaptation logic may be difficult due to design time uncertainty; *e.g.*, anticipating all potential environment changes at design time is, in most cases, infeasible. In addition, due to simplified design assumptions, the precise effect of an adaptation action may not be known, making it difficult to accurately determine how the SIS should adapt itself. The Online Learning Enabler addresses these challenges by leveraging modern machine learning algorithms during the operation phase [16]. In particular, the enabler uses reinforcement learning to address design time uncertainty by learning suitable adaptation actions through interactions with the environment at run time.

Security and Privacy Control and Monitoring enabler (S&P): This enabler is a one-stop solution for the near real-time monitoring and control of security- and privacy-related anomalies across multiple layers of Smart IoT Systems, from things, devices, Edge to Cloud. DevOps teams use the S&P enabler for controlling data protection and secure communications all along the lifecycle of the SIS, through continuous monitoring of security metrics, and automatic detection and feedback for subsequent DevOps loops. The enabler uses machine learning to correlate data captured by multiple probes or monitoring agents deployed in different layers, in order to offer a holistic view of the SIS and enable the detection of sophisticated attacks. The tool has a flexible architecture to adapt to the different information availability and the specific types of anomalies, and is fully elastic for the rapid scaling of the target systems. Further details about this enabler can be found in Chapter 6.

Context-Aware-Access Control (CAAC): Context-Aware-Access Control provides a unified access control of all the IoT actors from administrators, end-users, and services, to devices, and dynamically adapts the authorization according to the changing context [23]. It is a SaaS solution that can be integrated into the IoT applications, and provides a user-friendly authentication interface. The solution ensures the data is only exposed to authorized users and devices. It supports the applications in adapting the authorizations according to context changes, without requiring developers to modify the code. This is done by adding dynamicity to the OAuth 2.0 standard protocol to make the provided authorizations responsive to the context, injecting contextual risk levels as dynamic attributes in the authorization mechanisms. Further details about this enabler can be found in Chapter 6.

Root Cause Analysis enabler (RCA): Understanding the origin of a failure in a SIS is a complex and time consuming task. This is in particular due to the fact that these systems are large, vastly heterogeneous as well as widely distributed. This enabler observes the symptoms of the IoT systems, such as loss of messages, delay of response, etc. and automatically diagnoses the root cause, such as device failures or broken networks. DevOps teams can thus use the RCA enabler during the operation of their IoT application in order to receive alarms when there are incidents. The alarms will include details about the origin and possibly the reason of the accident as well as targeted instruction about how to fix the incidents. Instead of relying on human experts to exhaust all the causal connections between incidents and symptoms, the Root Cause Analysis tool builds this knowledge itself, by recording the typical incidents and their symptoms. During runtime, it compares the similarity between the observed symptoms with the recorded ones in the library to identify the possible incidents. Further details about this enabler can be found in Chapter 6.

2.1.4 Focus of the Different Enablers

In real cases, a large scale IoT system usually comprises many duplicates of the same or similar sub-systems, which contain a relatively smaller number of nodes. A typical example is the eHealth use case (see Chapter 9). In the eHealth use case, a remote patient monitoring system aims at supporting thousands of patients, and each patient is provided with a sub-system that includes one gateway and several sensors. These sub-systems are similar to each other, in terms of architecture, software and configurations. Under such setups, the DevOps of Smart IoT Systems usually includes two complementary activities: (i) the development, testing and optimization of the functionality within one sample sub-system, and (ii) the operation of the system of systems, with many duplicates of the sample sub-system.

The ENACT enablers naturally have different focuses. While the Risk Management, test and simulation, security and privacy control enablers and DivEnact are solutions that can be used at the system-of-systems level, the other enablers are aimed for the sub-system level. Yet, it is worth noting that the tools that system-of-systems enablers can also be applied at the scale of one sub-system.

2.2 Architecture of the ENACT Framework

The set of ENACT enablers introduced above form the ENACT DevOps Framework. Below we detail the architecture of this framework as well as the relationships between the enablers within this framework.

Figure 2.2 depicts the overall architecture of the ENACT Framework. It is a multi-layer architecture composed of 4 layers hierarchically organized plus one crosscutting layer. In the following we detail each of these layers. It is worth noting that Figures 2.1 and 2.2 are complementary in explaining the relationships

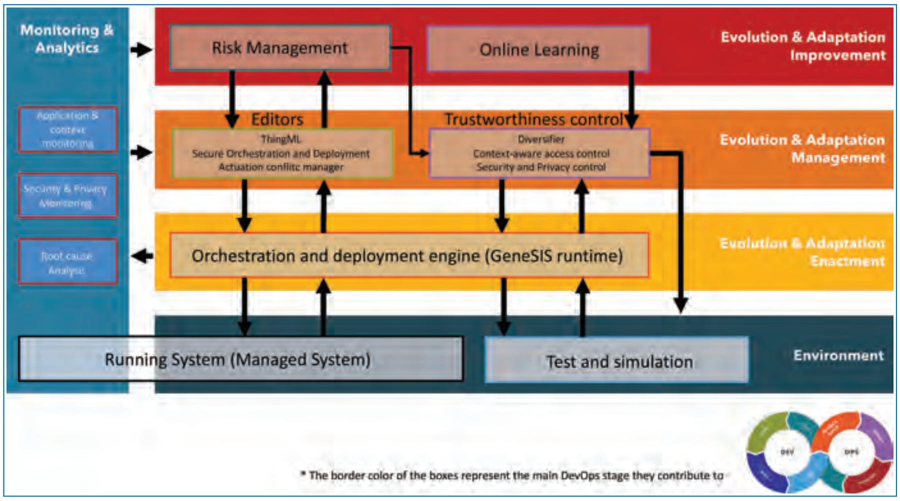

Monitoring & Analytics
Application & context monitoring
Security & Privacy Monitoring
Root cause Analysis
Risk Management
Online Learning
Evolution & Adaptation Improvement
Editors
ThingML
Secure Orchestration and Deployment
Actuation conflitc manager
Trustworthiness control
Diversifier
Context-aware access control
Security and Privacy control
Evolution & Adaptation Management
Orchestration and deployment engine (GeneSIS runtime)
Evolution & Adaptation Enactment
Running System (Managed System)
Test and simulation
Environment
* The border color of the boxes represent the main DevOps stage they contribute to
DEV
OPS

Figure 2.2. The ENACT architecture.

and complementarity of the enablers. The former details the contribution of the enablers within a DevOps pipeline while the later details how they can be integrated within a comprehensive Framework for the development and operation of SIS.

From the most abstract to the most concrete (*i.e.*, from the farthest to the closest to the running system), the layers are described as follows:

1. **Evolution & Adaptation Improvement Layer:** This layer provides the mechanisms to continuously improve and manage the development and operation processes of trustworthy SIS. On the one hand, the Risk Management enabler helps organizations to analyze the architecture of their Smart IoT Systems and detecting potential vulnerabilities and the associated risk (in particular related to security and privacy aspects) and propose related mitigation actions. On the other hand, the Online Learning enabler focuses on improving the behaviour of the adaptation engine that will support the operation of trustworthy SIS. This tool typically relates to the Operate stage of the DevOps process. In general, the improvement layer provides feedback and knowledge to all the other DevOps stages with the aim to improve the development and operation of trustworthy SIS. Thus, in this architecture, information from this layer are provided to the evolution and adaptation management layer with the aim to improve it.
2. **Evolution & Adaptation Management Layer:** This layer first embeds a set of editors to specify the behaviours as well as the orchestration and deployment of SIS across IoT, Edge and Cloud infrastructure. These editors integrate with mechanisms to maximize and control the trustworthiness of the

system. All together, these components cover activities in both the Dev and Ops parts of a DevOps process and in particular to the code, build and operate stages. The activities performed at this layer are strongly affected by the inputs from the improvement layer.

3. **Evolution & Adaptation Enactment Layer:** This layer bridges the gap between development and operation as its goal is to enact the deployment and adaptation actions decided at the Evolution & Adaptation Management Layer. The mechanisms of this layer monitor and manage the deployment of the running system.
4. **Environment Layer:** This layer consists of the running system together with the environment and infrastructure in which it executes. This includes both production and testing environments.
5. **Monitoring and Analytics Layer:** This layer is orthogonal and feeds the other four. The enablers at this layer are supporting the monitoring stage of the DevOps process and typically aim at providing feedback from Ops to Dev. More precisely, this layer provides mechanisms to monitor the status of the system and of its environment. This includes mechanisms to monitor the security and privacy of a SIS. In addition, it performs analytic tasks providing: (i) high level notifications with insights on ongoing security issues, (ii) diagnostics and recommendations on system's failures, and (iii) feedback on the behavioural drift of SIS (*i.e.*, system is functioning but not delivering the expected behaviour).

2.3 Improving SIS Trustworthiness

In this section we first summarize the contributions of the enablers in terms of supporting the development and operation of trustworthy SIS.

Based on the NIST definition of trustworthiness for Cyber Physical Systems [13], within ENACT, we adopt the following definition of trustworthiness and its different properties: "Trustworthiness refers to the preservation of security, privacy, safety, reliability, and resilience of SIS".

We adopt the following definitions of the different properties:

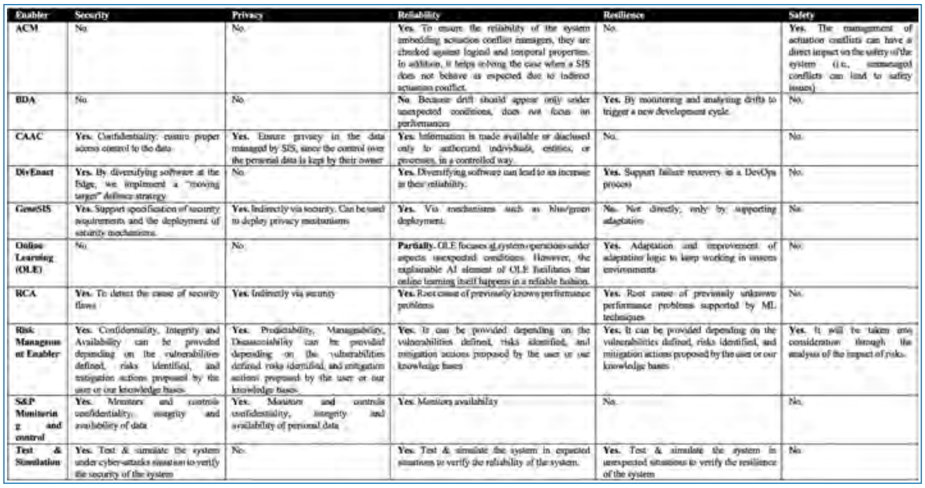
- **Security** refers to the preservation of confidentiality, integrity and availability of information [9].
  - **Integrity** is the property of protecting the accuracy and completeness of information [1].
  - **Confidentiality** is the property that information is not made available or disclosed to unauthorized individuals, entities, or processes [1].

  – **Availability** is the property of information being accessible and usable upon demand by an authorized entity [1].

- **Privacy** refers to the protection of personally identifiable information (PII) [10]. PII refers to any information that (a) can be used to identify the PII principal to whom such information relates, or (b) is or might be directly or indirectly linked to a PII principal.
- **Safety** refers to the ability of the cyber-physical system (CPS) to ensure the absence of catastrophic consequences on the life, health, property, or data of CPS stakeholders and the physical environment [13].
- **Reliability** refers to the ability of the CPS to deliver stable and predictable performance in expected conditions [13].
- **Resilience** refers to the ability of the CPS to withstand instability, unexpected conditions, and gracefully return to predictable, but possibly degraded, performance [13].

Figure 2.3 summarizes how each individual ENACT enabler contributes to the development and operation of trustworthiness of SIS. It is also worth noting that the support offered by the ENACT enablers to the DevOps of SIS is, by itself, a major contribution for supporting the trustworthiness aspect. Indeed, the adoption of the DevOps principles and practices in the field of the IoT is decisive to enable the continuous and agile evolution of SIS, which is necessary to adapt the system to newly appearing trustworthiness threats and to ensure its overall quality.

Some enablers are marked as indirectly contributing to the privacy property. This is because the support for security provided by these enablers also contributes



Enabler	Security	Privacy	Reliability	Resilience	Safety
ACM	No	No.	**Yes.** To ensure the reliability of the system embedding actuation conflict managers, they are checked against logical and temporal properties. In addition, it helps solving the case when a SIS does not behave as expected due to indirect actuation conflict.	No.	**Yes.** The management of actuation conflicts can have a direct impact on the safety of the system (i.e., unmanaged conflicts can lead to safety issues)
BDA	No	No.	**No.** Because drift should appear only under unexpected conditions, does not focus on performances	**Yes.** By monitoring and analyzing drifts to trigger a new development cycle.	No.
CAAC	**Yes.** Confidentiality: ensure proper access control to the data	**Yes.** Ensure privacy in the data managed by SIS, since the control over the personal data is kept by their owner	**Yes.** Information is made available or disclosed only to authorized individuals, entities, or processes, in a controlled way.	No.	No.
DivEnact	**Yes.** By diversifying software at the Edge, we implement a "moving target" defence strategy	No.	**Yes.** Diversifying software can lead to an increase in their reliability.	**Yes.** Support failure recovery in a DevOps process	No.
GeneSIS	**Yes.** Support specification of security requirements and the deployment of security mechanisms.	**Yes.** Indirectly via security. Can be used to deploy privacy mechanisms	**Yes.** Via mechanisms such as blue/green deployment.	**No.** Not directly, only by supporting adaptation	No.
Online Learning (OLE)	No.	No.	**Partially.** OLE focuses at system operations under aspects unexpected conditions. However, the explainable AI element of OLE facilitates that online learning itself happens in a reliable fashion.	**Yes.** Adaptation and improvement of adaptation logic to keep working in unseen environments	No.
RCA	**Yes.** To detect the cause of security flaws	**Yes.** Indirectly via security	**Yes.** Root cause of previously known performance problems	**Yes.** Root cause of previously unknown performance problems supported by ML techniques	No.
Risk Management Enabler	**Yes.** Confidentiality, Integrity and Availability can be provided depending on the vulnerabilities defined, risks identified, and mitigation actions proposed by the user or our knowledge bases.	**Yes.** Predictability, Manageability, Disassociability can be provided depending on the vulnerabilities defined, risks identified, and mitigation actions proposed by the user or our knowledge bases.	**Yes.** It can be provided depending on the vulnerabilities defined, risks identified, and mitigation actions proposed by the user or our knowledge bases	**Yes.** It can be provided depending on the vulnerabilities defined, risks identified, and mitigation actions proposed by the user or our knowledge bases	**Yes.** It will be taken into consideration through the analysis of the impact of risks.
S&P Monitoring and control	**Yes.** Monitors and controls confidentiality, integrity and availability of data	**Yes.** Monitors and controls confidentiality, integrity and availability of personal data	**Yes.** Monitors availability	No.	No.
Test & Simulation	**Yes.** Test & simulate the system under cyber-attacks situation to verify the security of the system	No.	**Yes.** Test & simulate the system in expected situations to verify the reliability of the system.	**Yes.** Test & simulate the system in unexpected situations to verify the resilience of the system	No.

Figure 2.3. ENACT contribution to SIS trustworthiness.

preserving the privacy of a SIS. The same applies to the safety property, the contributions of the enablers on security, privacy, reliability and resilience properties are important to help ensuring the safety of a SIS.

The enablers from the monitoring and analytics layer of the ENACT DevOps Framework (*i.e.*, security and privacy monitoring, behavioural drift analysis, and root cause analysis) are considering security, privacy, reliability and resilience aspects. It is worth noting that these tools are complementary: On the one hand, the security and privacy monitoring enabler focuses on observing symptoms of security and privacy issues, and the behavioural drift analysis enabler focus on symptoms of reliability and resiliency issues. On the other hand, the root cause analysis focuses on understanding the causes of these symptoms.

2.4 Evaluation and Validation: the ENACT use Cases

The general applicability of the ENACT enablers was validated and demonstrated in the context of three use cases: Smart Building, Intelligent Transport System (ITS), and eHealth. Each of these use cases represent different application domains, all facing specific trustworthiness challenges as depicted in Figure 2.4.

2.4.1 The Smart Building use Case

The first use case explored and validated ENACT in the domain of Smart Buildings, *i.e.*, Smart IoT Systems that make use of Smart Building sensors, actuators and services. The use case leveraged the Kubik test facility,[2] which is a three floors smart building owned by Tecnalia and designed for testing and research. Kubik

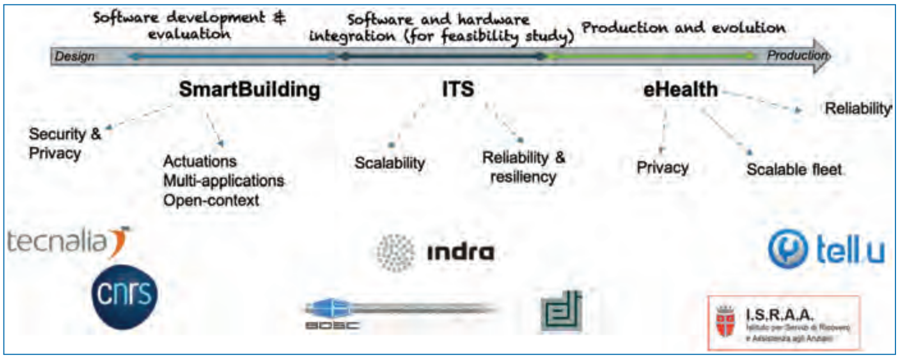


Figure 2.4. ENACT use cases and project partners in charge of the use cases.

2. https://www.tecnalia.com/images/stories/Catalogos/CAT_KUBIK_EN_dobles.pdf

offers SIS providers with a flexible framework not only to explore the opportunities offered by a rich ecosystem of sensors and actuators when designing novel IoT solutions but also to test and make experiments with the SIS resulting from this design in a real infrastructure. Thus, the smart building use case helped us validating our ENACT enablers in the early design stage of a SIS. During the project, several applications dedicated to aspects such as energy efficiency or user comfort were designed, developed and tested in Kubik. This context introduced specific DevOps and trustworthiness requirements that motivated the use of the some of the ENACT enablers.

For SIS providers it is especially important in this early design phase being able to quickly deploy and test the different applications and services that will compose or extend the existing SIS and thus run on IoT, Edge and Cloud infrastructure. The GeneSIS and TaS enablers aim at supporting DevOps teams in such activities.

Smart Building systems are typically composed of several applications controlling different actuation devices within the building (*e.g.*, HVAC, roller shutters, lights, TVs). In such a setting, it is of paramount importance to make sure the actuators are properly managed as to control their effects on the environment (*i.e.*, applications are behaving as expected). On the one hand, while it can be assumed that one application in isolation has a proper control over the actuators it applies, from the SIS perspective this assumption does not sustain as several applications may concurrently control shared actuators. Without proper mechanisms to handle such situation, the behavior of the actuator can quickly become unpredictable and possibly harmful. The ACM enabler aim at support the design of such actuation conflict handling mechanisms. On the other hand, indirectly, one actuator, possibly managed by an application, may hinder the effectiveness of another, managed by another application. Avoiding such loss of effectiveness is a complex task, which, without proper support, requires a deep analysis of the applications under operations. The BDA enabler propose to relieve developers from such a task, whilst the ACM enabler help mitigating the problem.

As in many other domains, smart buildings typically expose a broad attack surface and their security must not be an afterthought. The S&P Monitoring and control enabler provide a means to observe the security of the SIS and support security by design.

More details about the use case can be found in Chapter 11.

2.4.2 The Intelligent Transport System use Case

The second use case explored and validated ENACT in the domain of Intelligent Transport Systems, in particular exploring how SIS could be used for train integrity control. INDRA, as the system integrator, needs to continuously evaluate the subsystems with both software and hardware from their suppliers (*i.e.*, EDI and BOSC in this project), and adjust the design and implementation of their main services accordingly in order to maximize the integration of the subsystems. DevOps guarantees the effectiveness of the integration process, and provides real-time feedback to both the integrator and the suppliers as reference for subsequent development activities. The ENACT enablers were thus exploited in the use case at a stage where the focus was on understanding how best the hardware and the software can integrate, and if the integrated solution fits requirements for the solution to be scaled in production.

In particular, key challenges included to understand (i) how the software performs on the gateway and handles failures as well as (ii) how the overall SIS scales as, in the long term, the number of gateways and sensors is aimed to grow up to thousands of nodes.

To understand how the software performs on the gateway the first step was to actually deploy it. The use case exploited GeneSIS for this. The later was integrated as part of the Indra delivery pipeline, making sure that, when a new version of the software is ready, it is only deployed if the train is in a state where such maintenance activity is authorized. The second step was to monitor the system under operation and to report and analyse any failure. Because there can be many reason from which a problem in the software may originate, the use case leveraged the Root Cause Analysis enabler to guide DevOps engineers through a faster understanding of the problem.

For the testing of the solution at scale, building a testbed consisting of real devices was not an option as each individual gateway is already expensive. Instead, the approach selected was to build a hybrid testing environment combining a few real devices with simulated sensors and gateways. In such context, to make the tests as relevant and realistic as possible, the simulated devices must be able to replay real data from real scenarios as well as to inject erroneous data providing a means to evaluate how the system performs when operating properly and when error occurs. The Test and Simulation enablers perfectly fits these requirements and was a natural choice for Indra to evaluate their solution. The enabler was used to record real data from the system, simulate large infrastructure composed of several hundreds of nodes, and test the system accordingly, sometimes also simulating errors and attacks.

More details about the use case can be found in Chapter 10.

2.4.3 The eHealth use Case

The third use case explored a solution for remote patient monitoring and assistance that leverage a Personal Health Gateways (PHG) installed in patients' homes.

The PHG is at the core of the service as it integrates and controls various types of sensors and medical devices (e.g., blood pressure meter, fall detection sensors, glucose meter, video surveillance, indoor and out-door location tracking, etc.), and ensure that the right data are provided to the various stakeholders and to the integrated systems. The services running on the gateway needs to be customized to patient and family needs and requirements. This eHealth solution was partially developed within ENACT and is now in production with a large set of PHGs in production.

For eHealth systems, security and privacy are of paramount importance and compliance to GDPR and ISO 27001 is mandatory. As a result, risks must be carefully analysed and the necessary security and privacy mechanisms must be implemented on the medical gateway and secure communications with the Tellu Cloud platform need to be ensured. For instance, no data can be stored or processed on the gateways without strong gateway authentication and without enforcing a strong binding between the patient and the gateway. Before ENACT, no solution in the market fitted Tellu needs, preventing the migration of services to their medical gateway and thus hindering the full exploitation of the gateways. The Context-Aware Access Control enabler is the first solution to address this challenge. In complement, the Risk Management enabler provided Tellu with a mean to continuously perform the required risk analysis but also facilitating its reporting.

This eHealth solution was partially developed within ENACT and is now in production with a large set of PHGs in production. Each gateway should be configured to best fit (i) patient and family needs and requirements, and (ii) its operation context, including the set of sensors and medical devices connected to it. When dealing with a large fleet of gateways, the service provider (Tellu) cannot afford to operate and configure each Personal Health Gateway manually as this could easily overwhelm their operation teams, resulting in a service that is not scalable. The DivEnact enabler aim at addressing this challenge.

More details about the use case can be found in Chapter 9.

2.5 Conclusion

This chapter provided an overview of the ENACT approach to help the reader better apprehend the next chapters of the book. The focus was in particular on (i) the different enablers offered by ENACT for supporting DevOps for SIS, (ii) the ENACT Framework showing how the enablers may be combined, as well as (iii) the validation of these results in the context of realistic use cases from different IoT domains. Details about the enablers and use cases, such as scientific basis, implementation, application, and effect, can be found in the following chapters.

References

[1] Matt Bishop. “Computer security: Art and science. 2003”. In: *Westford, MA: Addison Wesley Professional* (2003), pp. 4–12.

[2] Nicolas Ferry *et al.* “Continuous Deployment of Trustworthy Smart IoT Systems”. In: *The Journal of Object Technology* (2020).

[3] Nicolas Ferry *et al.* “ENACT: Development, operation, and quality assurance of trustworthy Smart IoT Systems”. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer. 2018, pp. 112–127.

[4] Franck Fleurey and Brice Morin. “ThingML: A generative approach to engineer heterogeneous and distributed systems”. In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE. 2017. pp. 185–188.

[5] Anne Gallon *et al.* “Making the Internet of Things More Reliable Thanks to Dynamic Access Control”. In: *Security and Privacy in the Internet of Things: Challenges and Solutions* 27 (2020), p. 61.

[6] Thibaut Gonnin *et al.* “Actuation Conflict Management Enabler for DevOps in IoT”. In *10th International Conference on the Internet of Things Companion*. 2020, pp. 1–4.

[7] Edward R Griffor *et al.* “Framework for cyber-physical systems: Volume 2, working group reports”. In: (2017).

[8] Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 2010. ISBN: 860-1401501176.

[9] J ISO. “ISO/IEC 27000: 2012, information technology-security techniques-information security management systems-overview and vocabulary”. In: *International Organization for Standardization* (2012).

[10] J ISO. “ISO/IEC, 29100.2011 Information technology-security techniques-privacy framework”. In: *International Organization for Standardization* (2011).

[11] Victor Muntés-Mulero *et al.* Model-driven Evidence-based Privacy Risk Control in Trustworthy Smart IoT Systems”. In: (2019).

[12] NESSI. *SOFTWARE CONTINUUM: Recommendations for ICT Work Programme 2018+*. NESSI report. 2016.

[13] Alexander Palm, Andreas Metzger, and Klaus Pohl. “Online reinforcement learning for self-adaptive information systems”. In: *International Conference on Advanced Information Systems Engineering*. Springer, 2020, pp. 169–184.

[14] Gérald Rocher *et al.* "An IOHMM-Based Framework to Investigate Drift in Effectiveness of IoT-Based Systems". In: *Sensors* 21.2 (2021), p. 527.

[15] Hui Song *et al.* "Model-based fleet deployment of edge computing applications". In: *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. 2020, pp. 132–142.

[16] Antero Taivalsaari and Tommi Mikkonen. "A roadmap to the programmable world: software challenges in the IoT era". In: *IEEE Software* 34.1 (2017), pp. 72–80.

DOI: 10.1561/9781680838251.ch3

Chapter 3

Privacy Issues Control in Continuous Risk Management

By Victor Muntés-Mulero, Jacek Dominiak, Elena González-Vidal, Guillaume Mockly, Yuliya Miadzvetskaya and Tommaso Crepax

3.1 Introduction

In order to fully exploit the potential of IoT, it is crucial to facilitate the creation and operation of trustworthy Smart IoT Systems or, for short, trustworthy SIS. The different dimensions of trust for IoT systems were described by Yan et al. [22] concluding that risk management is an essential piece to guarantee trustworthiness. Markets in the need of trustworthy SIS, such as Smart Vehicles, Smart Grids or eHealth, are just flourishing and businesses will be continously adapting to new technologies. In this context, poor risk management together with a reactive strategy usually forces companies to continuously re-factor application architectures to improve software quality and security, incurring high re-implementation costs [2]. Besides, there is a lack of solutions to support continuous control of risks. In general, organizations struggle to collect valuable evidence to control on actual effectiveness of the mitigation actions defined during risk management process. In addition, many organizations use manual procedures based on using spreadsheets, by departments and locally [1]. This approach becomes quickly inefficient as projects or teams grow.

In the ENACT project, the Risk Management enabler is an evolution of the MUSA (H2020 Project No. 644429) Risk Management tool. While the tool created in the MUSA project focused in assessing risks and mitigation actions of Cloud Security based primarily on security related risks, ENACT enabler has evolved towards trustworthy SIS. While MUSA risk management tool explored risks related to the use of cloud services and recommended cloud services to be leverage from the system, ENACT enabler is able to consume the whole architecture of a SIS, expressed in GeneSIS, and find any type of vulnerabilities related to entities, processes, data store or data flows in the system. The basic risk management methodology the enabler is based on is described in [12] and, during the project, the risk management enabler has been effective embedded in the software development lifecycle both from a Dev and Ops perspective. Besides, although automated vulnerability detection is also discussed in [11], the details on how to create a knowledge base to support the detection are not discussed.

In parallel, GDPR discusses data protection by design and by default, remarking that it is essential to consider privacy from the beginning of any software development process to address related issues successfully. This is specially important for trustworthy SIS, since many IoT technologies are still under evolution and mixing legal requirements with a deep technical understanding is challenging. Therefore, including privacy aspects in a continuous risk management process is not straightforward.

Previous work related to this enabler, made in collaboration with the PDP4E project (H2020 Project No. 787034), is focused on showing how we embedded privacy-related risks explicitly through the combined use of models for both the architecture and the data flow implemented on the components of the architecture [12]. We achieve this by enabling the use of the information that is typically collected from the infrastructure to control security, thanks to the link that LINDDUN [21] establishes between privacy and security threats in STRIDE [16]. However, relevant aspects such as risk severity assessment is unclear, since most risk rating methodologies such as OWASP Risk Rating methodology are focused on security rather than privacy. As a consequence, impact assessment is usually focused on protecting the components of the system or the organization rather than data subject (DS) rights or other privacy-related principles.

Besides, even when it is possible to detect risks related to privacy from an engineering perspective (for instance based on LINDDUN) there is not a clear link between these and the main assets to be protected described in GDPR, including GDPR principles and DS rights. This makes it difficult for an organization to declare how they stand in front of GDPR in terms of risk control. While this issue is not exclusively relevant for IoT systems, it is essential to guarantee trustworthiness, specially when IoT devices are involved and the complexity of the architecture of the system grows, together with the attack surface. Because of this, we found it essential to guarantee an explicit analysis of privacy challenges in this book.

In this chapter, we focus on the missing pieces mentioned above, namely:

1. Extension of the impact assessment methodology to enable a more privacy-friendly assessment.
2. Discussion about the mapping of engineer-related aspects with higher level concepts in GDPR such as principles and rights.
3. Creation of a knowledge base to enable automated vulnerability detection.
4. Evaluation of the enabler in an IoT-based use case scenario related to smart vehicles.

This chapter is organized as follows: Section 3.2 provides a brief description of some previous work used through the chapter to found its main contributions. Section 3.3 proposes an extension to the OWASP Risk Rating methodology. Then, Section 3.4 discusses the relationship between LINDDUN threats and GDPR-friendly vocabulary related to data processing principles and DS rights. Section 3.6 describes an IoT use case related to smart vehicles where trustworthiness is essential and we discuss the concepts presented in the previous sections. Finally, we draw some conclusions related to the contributions of the chapter.

3.2 Previous Work

3.2.1 LINDDUN Methodology

LINDDUN is a threat modelling methodology that encourages risk analysts to address privacy risks affecting end-users of the application or system. This methodology provides some guidance to identify and categorize threats under a set of general risks (Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, and Non-compliance). LINDDUN is sometimes considered the privacy-oriented alternative to the STRIDE framework [16]. In fact, LINDDUN threats are described in the so-called LINDDUN trees which are explicitly connected to STRIDE threats trees. LINDDUN methodology requires to formalize the functionality of the system and its dependencies with respect to personal data. In such sense, LINDDUN proposed the usage of the Data Flow Diagrams (DFD) [4]. The notation of a DFD is based upon 4 distinct element types: (i) an external entity (i.e., end-users or third-party services that are external to the system), (ii) a data flow (explains data propagation and dependencies between all the functional components), (iii) a data store (i.e., a passive container of information) and (iv) a process (i.e., a computation unit).

3.2.2 Automated Vulnerability Detector

In order to facilitate an effective identification of privacy-related risks, it is important to make it easy for our enabler users to detect the vulnerabilities that expose the system to attacks that may violate DSs' rights. For that, we created an Automatic Vulnerability Detector (AVD) [11]. An AVD starts out from a set of DFDs to describe a software system under development. Based on these DFDs, it is able to detect potential vulnerabilities to kick off the risk analysis process. As explained, the AVD relies on a list of conditions that need to hold for a vulnerability to be effective. These conditions will need to be defined and stored in the Knowledge Base for the correct performance of the AVD.

3.2.3 Common Weaknesses Enumeration (CWE)

According to their website,[1] CWE™ is a community-developed list of software and hardware weakness types. It serves as a common language, a measuring stick for security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

3.2.4 Common Attack Pattern Enumeration and Classification (CAPEC)

According to their website,[2] CAPEC™ helps by providing a comprehensive dictionary of known patterns of attack employed by adversaries to exploit known weaknesses in cyber-enabled capabilities. It can be used by analysts, developers, testers, and educators to advance community understanding and enhance defences.

3.2.5 GDPR Enforcement Tracker

According to their website,[3] the CMS Law GDPR Enforcement Tracker is an overview of fines and penalties which data protection authorities within the EU

1. CWE – Common Weakness Enumeration (mitre.org): https://cwe.mitre.org/
2. CAPEC – Common Attack Pattern Enumeration and Classification (CAPEC) (mitre.org): https://capec.mitre.org/
3. GDPR Enforcement Tracker – list of GDPR fines: https://www.enforcementtracker.com/

have imposed under the EU General Data Protection Regulation (GDPR). Our aim is to keep this list as up-to-date as possible. The list does not list any fines imposed under national/non-European laws, under non-data protection laws (e.g. competition laws/electronic communication laws) and under "old" pre-GDPR laws.

3.3 Extending Risk Rating Methodology for Privacy

The ENACT Risk Management enabler uses LINDDUN as the baseline for privacy threat modelling. LINDDUN, just like any other modelling system based on STRIDE, has the issue that, once you automate the threat elicitation process, it returns as output an enormous amount of potential threats. Therefore, engineers need to identify in a given system what are, among a pool of many, the threats that actually need mitigation. At this point, we resort to risk assessment to prioritize the risks to mitigate.

Among the many risk assessment methodologies, the ENACT Risk Management enabler is based on the risk rating methodology of OWASP [20], a widely tested and accepted risk rating methodology for security. Unfortunately, the security nature of OWASP implies that the objectives it aims to achieve only partially intersect, but do not fully align, with those of privacy engineering. On the one hand, in 'traditional' security, the risk assessment is mainly carried out on behalf and benefit of the organization. Simply put, if the organization faces economic losses, the impact is deemed negative. Differently, in privacy and data protection, the assessment is made on behalf and interest of the DS, meaning that even if the organization can profit, the impact is negative if the DS suffers from a violation of its rights and freedoms. On the other hand, even though privacy engineering objectives of predictability, manageability and disassociability are in line with GDPR principles, we nonetheless acknowledge the existence of ontological differences between engineering objectives and privacy legal principles.

With all this in mind, the aim is to ensure that the use of OWASP does not undermine the protection of personal data. To do so, it is necessary to check up to which point OWASP's methods address legal requirements and, when needed, to customize them for privacy compliance.

3.3.1 Risk Appraisal and Risk Assessment

From a practical perspective, should a controller wish to process personal data, it is required by article 35, paragraph 1 GDPR to make two assessments. First, it has to assess whether the type of processing to be carried out is "likely to result in a high risk to the rights and freedoms of natural persons", which we call "Risk Appraisal".

Should the outcome of the Risk Appraisal be positive, then a second assessment is in order, this time on the "impact of the envisaged processing operation" – also known as Data Protection Impact Assessment, or DPIA. The Article 29 Working Party released official guidelines on how to conduct both [13]. It shall be noted that, both stages consider the overall risk value from the perspective of risk analysis (i.e. encompassing both what we term as 'likelihood' and as 'impact', regardless the different wording employed by the GDPR), albeit the former does so in a shallower and more abstract way.

3.3.2 A GDPR-friendly, OWASP-Based Privacy Risk Estimation System

DPIAs and Risk Appraisals are functionally dependent on privacy risk assessments. For this reason, we thought it would be twice as useful to propose an effective privacy risk rating system to underpin either of them. For the privacy risk rating system to be GDPR friendly, we look into what the GDPR requires in regards to DPIAs and Risk Appraisals and extrapolate concepts to use as factors.

The law is not clear in determining whether the concepts that are critical to the initial Risk Appraisal and the risk assessments are different. For example, recital 84 GDPR states that aspects to consider for risk evaluation are origin, nature, particularity and severity, but does not clarify whether such aspects only relate to risk assessment or also to Risk Appraisal. In addition, the WP29 is of the opinion that controllers have a constant obligation to implement measures to manage privacy risks:

> 'The mere fact that the conditions triggering the obligation to carry out DPIA have not been met does not, however, diminish controllers' general obligation to implement measures to appropriately manage risks for the rights and freedoms of DSs. In practice, this means that controllers must *continuously* assess the risks created by their processing activities in order to identify when a type of processing is "likely to result in a high risk to the rights and freedoms of natural persons".

The ambiguity of the law on one side, and a more functional approach towards risk assessment on the other, not only seem to allow for, but to encourage that risk management be continuously active in parallel to the data processing activity. In our case, this translates into the chance to use the same tool for Risk Appraisal, risk assessment and even to check whether there are residual risks after the DPIA is conducted. Consequently, since our study provides for a more granular analysis of privacy risks, it can discover issues at earlier stages of the process, and is partially automated, it can be used repetitively by the data controller to track and manage changing privacy risks over time.

The GDPR key in relation to DPIAs is article 35 paragraphs 1, 3 and 4, together with a number of recitals giving insights on what the law considers important to determine the severity of a risk, namely 71, 75, 76, 84, 89, 91, 92, and 116. By a combined reading of article 35 and the recitals, the WP29 extrapolated 9 processing operations as 'likely to result in a high risk' for the DS. If two or more of the following coexist, then the high risk is likely to occur and, thus, a DPIA is in order.

The processing operations are:

1. Personal evaluation or scoring of the DS, including profiling and predicting;
2. Automated decision-making that significantly affects the DS;
3. Systematic monitoring that results in observation, monitoring, or controlling of DSs;
4. Processing of sensitive or highly personal data;
5. Data processed on a large scale, considering number of DSs, volume and range of data, duration of activity and geographical extent;
6. Matching or combining data-sets;
7. Vulnerable DSs, when there is a power imbalance between the controller and the subject who is unable to consent or object to the processing;
8. New technology or innovative use of technology or organizational solutions;
9. Processing prevents a DS to exercise its rights, enter into contracts or make use of services.

Rather than systematizing privacy risk assessments, the GDPR gives a number of rules scattered among articles and recitals on how to understand what to consider while evaluating the severity of privacy risks. Similarly do the Guidelines of the WP29, which only better refine the categories of data processing operations considered 'high risk'. Therefore, one has to resort to the privacy engineering academic scholarship to find attempts to systematize privacy risk assessments that can help quantifying privacy risk factors.

Methodologically, the difficulty that any expert encounters when estimating risk values depends on that their factors, namely likelihood and impact, are impossible to quantify with precision. Only a few scholars have tried to lay the theoretical foundations for such assessment, and it is in fact from the studies of the building blocks of privacy risk metrics by Wagner and Boiten 2018 [19] that we start our exercise of combining the requirements of the GDPR, their interpretations by the WP29, and OWASP risk rating.

Our aim is to model a privacy risk rating system on the basis of the data processing operations considered 'high risk' by the WP29, with the further trust that such system will guarantee a high level of compliance with GDPR requirements.

In the next sections we put forward our solutions to address the issues of estimating risk likelihood and impact.

3.3.3 Likelihood

The calculation of likelihood is one that risk methodologies take at best as rough estimate, mostly because risks may or may not materialize due to a number of unforeseeable circumstances, as well as their probability of occurrence being stretched over an uncertain amount of time. Moreover, it is hard to determine complexity, variation and hiding of multiple root causes and consequences associated to each risk.

The imprecision of likelihood measurements does not put the privacy risk assessment to a halt. In fact, from a functional perspective, risk severity — labelled on a scale "from low to high" — provides enough data to inform risk management decisions in compliance with GDPR requirements. Nevertheless, a more accurate quantification of likelihood is important because the privacy controls that will be used for the mitigation of privacy threats will most likely decrease risks' likelihood, rather than impact [19].

The OWASP likelihood estimation methodology considers two sets of factors, the first being *threat agents* and their characteristics, and the second being *vulnerabilities*. Different threat agents, or attackers, are analysed on the basis of their potential *skills*, *motives*, *opportunities* and *size*. The idea behind such differentiation is that, for instance, attackers coming from the inside of an organization may have more opportunities in terms of access than outside attackers, yet be less skilled in terms of hacking abilities.

Privacy and security risks are different in nature, but the analysis for determining their likelihood seems, at first sight, similar. In fact, the determination of likelihood is only similar for those privacy risks that share analogous characteristics with security risks. Consequently, such privacy risks' likelihood is rated on the basis of how easily can a vulnerability be discovered and exploited by an identified threat agent, how many threat agents of the same type know about the vulnerability (i.e., awareness), and what intrusion detection measures are put in place against exploits by threat agents. Visibly, OWASP's determination of likelihood is fundamentally connected to threat agents, fact that depends on OWASP being designed on security attacks.

Regrettably, what OWASP does not consider is that threats may not be caused by a willing threat agent. In fact, there are privacy risks that lie outside the attacker-type scheme. As far as data protection is concerned, the controller organization itself can be considered as an attacker from which the DS shall be protected. Upon this assumption, many privacy-by-design and minimization concepts are rooted.

Bearing in mind the mentioned difference between 'traditional' security and privacy risk assessments, back to the comparison with OWASP, the threat agents in the privacy case are still the same individuals as in security, that is organization employees, executives, etc.; however, for a given risk, they will have different motives, such as the exploitation of the DSs' personal data for economic advantage – more a matter of privacy than security [10].

Harms, both for security and privacy, can be caused by a poorly designed policy within an organization, the careless work of a DPO, or even the use of a badly designed tool for risk estimation. All these events increase the likelihood of materialization of adverse effects on the rights and freedoms of the DS, which the NIST defines "problematic data action", an "operation that a system is performing on personally identifiable information, that could cause an adverse effect or a problem for individuals [3].

Accordingly, the likelihood of problematic data actions cannot be quantified just over the characteristics of what may not be an attack. Therefore, the NIST suggests that, within a specific context, controllers take DSs' perceptions of which data actions they consider problematic through customer demographics, focus groups, surveys, etc. Once a list of problematic data actions is created, it should be possible to determine the likelihood of their happening. If, for instance, in one specific area, DSs have indicated "destruction of personal data due to earthquake" as problematic data action, the controller should be able to determine the likelihood of an earthquake happening. Similarly, if the DSs have identified "ambiguity of privacy policy wording", a controller should be able to register how many times did such unwanted events happen in its organization. Such problematic data actions can be monitored and quantified, and with them their likelihood.

A rating can be created to determine whether data actions that are perceived as problematic happen in the real world in a fashion that is rare to unlikely (1 to 3), possible to likely (4 to 6), or almost certain to certain (7 to 9). The mentioned levels mimic those of OWASP, where the likelihood of a security risk happening is rated as low (1 to 3), medium (4 to 6) or high (7 to 9).

3.3.4 Impact

In comparison to security, it is the use of OWASP to quantify the impact of privacy risks on DSs that presents the most substantial differences. Such differences, in turn, imply equivalent adjustments to the privacy risk rating system. Keeping the framework of OWASP as baseline, we combine it with the impact factors, categories and dimensions of Wagner and Boiten, mentioned before. To every impact category of Wagner and Boiten, namely, harm, scale, sensitivity and expectation, we map the key aspects of WP29's processing operations. To appreciate the varying impact of each of the four categories, we will do a simple exercise of analysing one category while keeping the other three constant.

OWASP divides the impacts of an attack into two categories, namely 'technical impact' on application, data, and functions, and 'business impact' on the organization. In regards to technical impacts, OWASP lists the loss of confidentiality, integrity, availability and accountability as factors. Evidently fundamental to security, such factors also have repercussions on privacy so long as the confidential, incorrupted, available and accountable information are personally identifiable. This means that, the four technical factors in OWASP for privacy are similar to, but have a much restricted material scope that excludes all data other than personal.

Harm. In regards to business impacts on the organization, it is crucial to understand that "only individuals — not agencies can *directly* experience a privacy problem" [3]. This means that each individual has a different perception of the harm caused by one problematic data action, and that such perception may also vary depending on the context.

The most important consequence of the personal nature of impacts is that it makes them very challenging to quantify consistently. NISTIR 8062 does not address the problem of quantification of harms directly, but suggests instead that businesses (or organizations) use costs, such as reputational or legal costs incurred for legal compliance, as proxies for the quantification of individuals' impacts. Wagner and Boyten suggest a different solution, that is either using a Likert scale (called 'perceived harm') or, as a proxy, the amount of damage that a court would be likely to grant (called 'damages awarded') [19].

Although non-optimal, the measure of harm from the standpoint of a DS is arguably to average scales similar to Likert's. An organization willing to understand the perceived harm to the DS involved in its systems should answer the following questions: "How much do you think that this problematic privacy action would harm the DSs related to your system? Not at all to moderately (0 to 3), considerably to significantly (4 to 6), highly to irreparably (7 to 9)".

The scales solve the problem of defining and finding a common metrics to harms of different nature, such as reputational harm, financial harm, etc. We suggest organizations to conduct a survey with their DSs to get a better understanding of how much the dreadful event would impact them. The averaging of the Likert scales comes to solve the problem that harm is felt differently among DSs, and it is thus impossible to tailor its measuring to each DS involved in a problematic data action.

It is safe to say that all high risk operations can be mapped to the category of harm. This is due to that, if no harm were to be inflicted to the DS, the related risk would not exist. We can take as examples the following categories: automated decision-making that significantly affects the DS, because it may bring to discrimination and exclusion, which are harms that are personally felt differently from one DS to another; systematic monitoring, because the knowledge of being constantly monitored is also perceived differently by different DSs, and may affect their behaviour accordingly; vulnerable DS, because the power imbalance between the controller and the DS is greater when the latter is a child, an employee, a mentally ill person, an elderly, an asylum seeker, a patient, or another category of people who are unable to consent or oppose to processing due to relational or personal circumstances; processing prevents a DS to exercise its rights, because, e.g., the inability to enter into an insurance contract has different implications depending on the denied person, who not only may personally perceive the denial differently, but also be objectively awarded different damages by a court depending on the circumstances of the case.

Scale. To Wagner and Boiten, between two problematic data actions that affect (a) the same type of data (e.g., medical data), which belong to (b) equally harmed DSs (that is, DS who would feel the same personal harm as well as would be awarded the same amount of damages by a court) with (c) the same expectation of privacy, the one with the greater impact is that which affects the larger number of people.

Thanks to the processing operation 'data processed on a large scale', we are able to extend the scale category to a second dimension that is, volume of data. As regards to volume, the processing of more data items has a bigger impact than that of fewer data items: considering two data-sets, A and B, which contain exactly the same personal data belonging to the exact same people, the action of replicating multiple times and in different places data-set A would have a bigger impact on the DSs, because the chance for unlawful processing is likewise multiplied, or because more personal data are anyhow more demanding to protect.

Measuring scale is perhaps the easiest quantification among the impacts categories, because the number of DS involved and the volume of data are all objective, ordinal numbers that are either known, or so can be through data analytics. It is possible to use a specific category in OWASP called 'Privacy Violation' that combines the dimensions of volume and number of persons by measuring how much personally identifiable information could be disclosed by one particular processing activity. OWASP lists a number of options, and gives to each option an impact rating (in brackets), from 0 to 10: one individual (3), hundreds of people (5), thousands of people (7), millions of people (9).

Sensitivity. Keeping other impact categories constant, the more sensitive the processed personal data, the higher the impact on the DS. The law gives exceptional attention to data that, because of their nature, are considered special, namely: data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership; genetic data and biometric data used for the purpose of uniquely identifying a natural person; data concerning health; data concerning a natural person's sex life or sexual orientation; and data related to criminal convictions or offenses.

Additionally, the WP29 lists a number of data types that should be considered sensitive because they increase the risk to rights and freedoms [13] (Sensitive data or data of a highly personal nature): "personal data linked to household and private activities (such as electronic communications whose confidentiality should be protected), or because they impact the exercise of a fundamental right (such as location data whose collection questions the freedom of movement) or because their violation clearly involves serious impacts in the DS's daily life (such as financial data that might be used for payment fraud)".

Processing operations involving all such data are considered 'high risk' but, unfortunately, there is no way to objectively determine which of these special data types have a bigger impact on the DS without considering the context and purposes of use. However, on the one hand, the law gives sensitive data a greater weight compared to non sensitive personal data and, on the other, it is safe to say that, between two processing activities of the same volume about the same person, that which includes the most categories of special data types must have a bigger impact. For these reasons, Wagner and Boiten suggest to use the number of different data types as means to measure sensitivity.

One measuring rating for sensitivity could be created by answering the question: how sensitive is the processed personal data? The options, with related impact rating in brackets (from 1 to 10), could be: not in the list [13] of sensitive data types (2); Not in the list, but could be easily used to predict sensitive data (5), Matches 1 category in the list (7), Matches 2 or more categories in the list (10).

Expectation. DSs have reasonable expectations about how their personal data will be handled by a controller. For instance, when consent is given as legal basis for processing, a DS should be able to predict what will happen to its data; similarly, a DS managing privacy settings to decide what types of cookies is a website allowed to use, or what information can it share with third parties, has an expectation on that only those cookies will be stored, and only those specific information be shared with predetermined third parties. Once the expectation is set, it is possible to determine to what extent has the actual processing deviated from it.

Processing operations involving evaluation or scoring of DSs are generally precursory to profiling, or to some forms of behavioural prediction. They are considered high risk because often leading to one or more of the other high risk processing operations, such as discriminating DSs on the basis of their personal vulnerability, race or other sensitive data, automated decision-making significantly affecting the DS, or preventing DSs to exercise rights or enter contracts. For this reasons, between two collections of personal data, the one collection based on which a profile is created has a bigger impact on the DS.

Systematic monitoring of DSs with the purposes of observing, monitoring and controlling has a different impact on each DS. People tend to change their behaviour according to whether they know of being constantly monitored (so called 'chilling effect' [18]), and governments as well as private companies exploit more or less obtrusive technologies as means of control. When systematic monitoring is undetectable, personal data may be collected in circumstances where DSs may not be aware of who is collecting their data, how they will be used, and that personal data is being collected in the first place. When technologies for systematic monitoring are purposefully non obtrusive, the expectation of privacy of the DSs are very high and, thus, any type of personal data processing inherently diverges from such expectation.

Matching or combining data-sets of an unaware DS is an intrinsic violation of the principle of purpose limitation. Given a specific set of personal data, the DS should always be able to predict the consequences of a specific type of processing. The combination of multiple data-sets, thanks to data analytics, can reveal personal information that were not deemed to be shared within the principal processing, or even create new personal data [9]; both of the outcomes exceed the DS's expectation of privacy.

DSs have expectations on how a technology or a process will manage their personal data given the information they have on that technology at the time of collection. Therefore, innovative uses, or new technological or organizational solutions for data processing exceed such expectations unless the DS was put in the position to agree on the new means of processing. Given two processing operations on the same data of the same DS, the one using new technologies or solutions has a bigger impact.

To quantify the impact of exceeded expectation it is critical to first set a baseline and, to do so, we welcome Wagner and Boiten's suggestion to use Solove's taxonomy of privacy [17]. Based on the typically American concept of expectation of privacy, Solove's taxonomy is useful to determine what a DS expects from a data processing activity from the moment of collection, to dissemination, through management and storage. The divergence between the expected and actual means of processing, expected and actual types of created and shared data, and expected and actual consequences of processing can be measured by counting the number of exceeded categories of processing (collection, storage, dissemination, etc.) or, more granularly, by referring to the metrics we already used in other impact categories. This means that, for exceeded expectations on the types of processed data, one can refer to the higher sensitivity of the personal data, their bigger volume, the more severe personal or objective harm, and so on.

Another way to conduct such measuring is by considering that, as a general rule, the deviation from expectation gets bigger every time that the personal data, collected for a specific purpose, are reprocessed, re-used, re-analyzed, re-combined, etc. However, an engineer may not be able to count that, as the code may be implemented by several people. Therefore, we follow Solove's taxonomy and focus on expected intrusiveness into DS's life through the following question: "Considering that a potential system may collect, analyse, process and disseminate information, what is, in the eye of a DS, your system expected to do with the information?". Collect, analyse, process and disseminate information (2); Only disseminate information (4); Process, without disseminating information (6); Only collect information (9)". The idea is, the less the user expects the system to do with the information, the farther it will be from their expectation if a breach happens.

3.3.5 Summing up

To assess likelihood, a controller could determine in what fashion do data actions that are perceived as problematic by a DS happen in the real world. To assess impact, there are 4 categories to consider: harm, calculated on the controller's estimation of the damages to its specific categories of DSs; scale, calculated on the basis of how many individuals are involved in the processing activity; sensitivity, measured depending on whether the processed data belong to one or more of the categories identified by the WP29 as "high risk"; and expectations, calculated on the (un)expected intrusiveness of the data processing into a DS's private life, from the perspective of the data controller.

3.4 Connecting Engineer-driven Privacy Practices with GDPR

Despite the fact that the GDPR is a legal text and LINDDUN is an engineering method, an attempt can be made to align LINDDUN and the GDPR in order to bridge the existing gap between legal and technical practices and, thus, address the demands of privacy engineering community of, first, translating legal jargon of rights, values and principles into notions and tools that engineers are more familiar with, such as threat trees, data flow diagrams, etc.; and second, of operationalizing the GDPR, particularly in the the detection of the privacy threats and the elicitation of the associated mitigation strategies.

3.4.1 Initial Considerations

We start by remarking some initial considerations about the relationship between LINDDUN and GDPR.

Linkability (L), identifiability (I), detectability (D), and to some extent non-repudiation (Nr) are all pointing out to the existence of personal data, since the occurrence of one of these threats could lead to the identification of a natural person. According to the European legislation, the anonymous information does not require protection for compliance with the principles of data protection. Anonymous data do not relate to an identified or identifiable natural person and are therefore considered non-personal. However, "in this era of big data, full anonymity is hard, if not impossible, and even more advanced anonymity techniques cannot guarantee full anonymity when data are linkable" [8]. The threat of linkability may necessitate a further analysis since it cannot be established without context whether the linkability of two items of interest would allow the identification of a natural person and, thus, qualify as personal data.

Linkability might lead to identifiability (i.e. linking data to an identity). Once the DS is identified or is identifiable, the information qualifies as personal data and triggers the applicability of GDPR.

Information disclosure links to arguably all principles of GDPR art. 5. In fact, when personal data are disclosed to non-authorised parties they are no longer under the control of the DS nor of the responsibility of the controller/processor, which means that all the procedural and substantial safeguards provided by art. 5 and related rights are exposed to risk of violation. Personal data shall be processed in such a manner that ensures appropriate security, including protection against unauthorised or unlawful processing, alteration or accidental loss.

Unawareness is linked to principles related to information requirements, as well as to the procedural enjoyment of the DS rights. Not only shall the DS be given all the information about data processing activities, but more importantly she has to be made aware that any processing of her personal data is happening. Unawareness links to the principle of lawful processing, insofar as the DS cannot consent to processing she is unaware of; same applies to any other right she is entitled to enjoy by active personal request (e.g., right to information, access, rectification, erasure, etc.).

Non-compliance threat could be associated with data protection by design requirement, accountability obligation under Article 24 GDPR, such as adopting appropriate technical and organisational measures ensuring the GDPR compliance or adopting internal privacy policies. For the most part we can speak about general GDPR non-compliance resulting in a pyramid of sanctions: from warnings to sanctions as a last resort.

In this subsection, we provide the description of each LINDDUN threat category and its relation with the GDPR.

3.4.2 Linkability

Linkability means "being able to sufficiently distinguish whether 2 IOI (items of interest) are linked or not, even without knowing the actual identity of the subject of the linkable IOI". Pfitzmann and Hansen give the following definition: "unlinkability of two or more items of interest (IOIs, e.g., subjects, messages, actions, etc.) from an attacker's perspective means that within the system (comprising these and possibly other items), the attacker cannot sufficiently distinguish whether these IOIs are related or not" [15]. For instance, unlinkablity of a message sender/recipient to a message sent or received or unlinkability of a relationship between a sender and a recipient [15]. Unlinkability is one of prerequisites of anonymity. Nevertheless, failing unlinkability will not necessarily eliminate anonymity, but will decrease its strength [15].

From a legal perspective, linkability means that the failure to hide the link between different actions, identities or pieces of information could potentially result in the unexpected personal data processing. For instance, the Article 29 WP provides for the following example: Titius has these fingerprints, this object has been touched by someone with these fingerprints and these fingerprints correspond to Titius, therefore this object has been touched by Titius [14]. Thus, linkability allowed to establish a link between one piece of information and the individual. The linking of different pieces of information can result in the misuse of the personal data by third parties. Such misuse can be caused by the failure to implement the necessary controls to ensure an appropriate level of protection of personal data (e.g., failed anonymization). If the controller is not aware of the personal data processing operation due to the failed anonymization, it will not be able to comply with the GDPR data processing principles and, thus, will fail to ensure the respect for DSs' rights. Thus, linkability may result in the violation of a number of the personal data processing principles and of DSs' rights listed in the GDPR.

Relationship with personal data processing principles. First, the principle of lawfulness will be violated since there will be no lawful grounds for processing, as provided in Article 6 of the GDPR. Second, the principle of transparency will not be complied with, because DS will not be informed about the processing activities over their data. The DS might not be even aware at all that such personal data have been collected, used, consulted or otherwise processed and what is the extent of this processing.

Third, the purpose limitation principle will be also jeopardized since the controller, unable to establish the existence of personal data, will not be able to ensure that the data collection is limited to "specified, explicit and legitimate purposes" (Article 5(1)(b) GDPR). Moreover, in this case the controller will be collecting personal data without knowing itself how and when these data will be used, since in its system the data are not identified as personal.

Moreover, the data minimisation and storage limitation principles will be also violated since the unawareness about the personal data mere existence will prevent controllers from establishing whether the same purpose can be achieved with a narrower amount of personal data and for a shorter retention period.

The inability to establish that the personal data exist in the system or that a third party can establish links between different pieces of information and, consequently, guess the existence of such data, will prevent us from ensuring that the data are accurate and kept up to date. As a result, controllers will not be able to ensure accuracy at all stages of collecting and processing of personal data and take every reasonable step to ensure that inaccurate data are erased or rectified without delay. Thus, contrary to the principle of accuracy, controllers will not make sure that outdated data are eliminated, or that data are correctly interpreted.

The compliance with the principle of integrity and confidentiality will be also jeopardized since the processing of the data, deemed as non-personal, will not be as secure as required for the personal data processing, "including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures" (Art. 5(1)(f) GDPR) . This will result in a lack of appropriate controls to prevent unauthorised access to the personal data as well as implement systemic quality controls in order to ensure that an appropriate level of security is reached. Moreover, the personal data will not be validated (e.g. using hashes), which might lead to some negative consequences, such as inability to guarantee its integrity and, consequently, the accuracy of that data.

According to the principle of accountability, the controller shall be responsible for, and be able to demonstrate compliance with, principles relating to processing of personal data and listed in Article 5 of the GDPR. The non-respect for one of these principles will trigger the accountability obligation.

Relationship with DS rights. Since linkability in many cases is undetected because the personal data is not recognized as such and is not traceable in the system, the controller will not comply with information obligation, as substantiated in Articles 13-14. Thus, DSs will be deprived of the right to obtain information about the processing activities over their data, the identity and the contact details of the controller, the purposes of the processing, the categories of the data and their recipients, and how the data are being controlled, monitored or used further. The information obligation is the essential first step setting out the stage towards the exercise of other DSs' rights. Neither right of access, nor right to rectification or erasure of personal data, nor restriction or objecting to their processing will be possible unless the DS knows the personal data is processed by the controller.

3.4.3 Identifiability

“Identifiability of a subject from an attacker’s perspective means that the attacker can sufficiently identify the subject within a set of subjects.” [15] Identity can be explained and defined as the opposite of anonymity and the opposite of unlinkability [15]. In a positive wording, identifiability enables both to be identifiable as well as to link IOIs. The less is known about the linking to a subject, the stronger is the anonymity. The anonymity decreases with a growing linking [15].

The definition of identifiability provided in the technical literature seems not to be totally in line with the legal understanding of an identifiable natural person. While both the legal and technical literature recognise pseudonimisation as one of the techniques decreasing the likelihood of identifiability, the GDPR takes a stricter stance on pseudonimised data. For instance, Recital 26 GDPR sets out that “pseudonimised personal data, which could be attributed to a natural person by the use of additional information should be considered to be information on an identifiable natural person”. And, thus, such data will be treated as personal under the GDPR, since pseudonym means that it is possible to backtrack to the individual and discover individual’s identity. At the same time, the technical literature admits the flawlessness and high linkability potential of pseudonimised data, but still seems to treat pseudonimity as a concept in a slight opposition to identifiability [8]. “Whereas anonymity and identifiability (or accountability) are the extremes with respect to linkability to subjects, pseudonymity is the entire field between and including these extremes” [21].

In addition the concept of identifiability is not that straightforward. For instance, the GDPR provides a non-exhaustive list of identifiers in Article 4, such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person. “The natural person is “identifiable” when, although the person has not been identified yet, it is possible to do it” [14]. But the likelihood of identifiability should be analysed on a case-by-case basis. For instance, a very common name will not necessarily allow to single out one particular person from the whole of a country’s population [14], but can achieve the identification of a pupil in the classroom. In addition, the name, combined with some additional information can also allow the identification of someone as a result of this “unique combination” set. Even a very descriptive information can identify someone, i.e. someone wearing a red hat at the bus stop at a particular moment.

The identifiability is a dynamic process and, while it may not be possible to identify someone today with all the available means, it may happen at a later stage due to a technological progress. To determine whether an individual is identifiable, Recital 26 GDPR underlines, "account should be taken of all the means reasonably likely to be used, such as singling out, either by the controller or by another person to identify the natural person directly or indirect". The likelihood of identification must be assessed in light of "objective factors, such as the costs of and the amount of time required for identification, taking into consideration the available technology at the time of the processing and technological developments".

Since identifiability is closely related to linkability, it will affect the same GDPR principles and DSs' rights.

3.4.4 Non-repudiation

Non-repudiation is the opposite of plausible deniability. Plausible deniability from an attacker's perspective means that she cannot prove a user knows, has done or has said something [21]. While the goal of non-repudiation is to provide irrefutable evidence concerning the occurrence or non-occurrence of an event, it must be admitted that some participants may desire that there is no irrefutable evidence concerning a disputed event or action [21]. Wuyts provides for some concrete examples where non-repudiation is a privacy threat. For instance, e-commerce applications, where the vendor can later use the signed receipt by the buyer as evidence that the user received the item. For other applications similarly users may desire plausible deniability in order to ensure that there will be no record to demonstrate the communication event.

In an attempt to single out the most linkable GDPR principles with non-repudiation, we came to the conclusion that non-compliance with integrity and confidentiality requirements might lead to the loss of control over the personal data and increase the probability that unauthorized parties can access it. Logically, the controller will be held accountable for such incidents and for lack of appropriate confidentiality strategies. We consider that right to be forgotten and right to rectification are intrinsically linked with plausible deniability, since they allow for ex ante rectification of the personal data inaccuracies and the possibility to ask for erasure of those data, which are no longer necessary for the purposes for which it was collected or where such purpose ceases to exist, or where the DS withdraws consent on which the processing is based. Thus, right to be forgotten and right to rectification will prevent a priori the third parties from getting access to the information, which the DS considers as inaccurate or compromising. Nevertheless, as provided in Article 17 GDPR some exceptions might apply to the exercise of the right to erasure, including the situations where there is a need to strike a right balance between public interests, freedom of expression and other competing rights and legitimate interests. In addition, Deng et al. notes with regard to plausible deniability that it ensures that "an instance of communication between computer systems leaves behind no unequivocal evidence of its having taken place" [5]. Thus, in relation to the right to be forgotten and right to rectification, one might ask whether the controller should store requests for personal data erasure or rectification. And would not such storage be detrimental to plausible deniability? Thus, the right balance should be again struck between accountability obligations and DSs' legitimate interests.

In addition, in order to guarantee plausible deniability the data controller may decide to make the data less accurate to "cover user's tracks". While the GDPR requires to keep the personal data up to date and ensure that inaccurate data are erased or rectified without delay, plausible deniability may require a different approach towards accuracy. On one hand, the accuracy of personal data should not be compromised, on the other hand, making personal data less discernible from the outside may be necessary for ensuring plausible deniability.

3.4.5 Detectability

"Undetectability of an item of interest (IOI) from an attacker's perspective means that the attacker cannot sufficiently distinguish whether it exists or not. If we consider messages as IOIs, this means that messages are not sufficiently discernible from, e.g., random noise" [15]. The difference between unlinkability and undetectability is the following: in unlinkability, the IOI itself is not protected, but only its relationship to the subject or other IOIs is protected. For undetectability, the IOIs are protected as such [21]. Undetectability consists in, for instance, hiding the user's activities or location [21]. Undetectability in the past was referred as unobservability. However, since unobservability comprises both anonymity and undetectability, LINDDUN method focuses on undetectability.

Detectability threat is strongly related to the context. It is impossible to establish without further details whether detectability of one particular activity can lead to identifiability of an individual. But if we assume that detectability results in an identifiability of a natural person, the scope of the GDPR will be triggered in a similar way to linkability and identifiability.

3.4.6 Disclosure of Information

Information Disclosure is the exposure of information to individuals who are not supposed to have access to it. Principles of integrity and confidentiality will be the most relevant to guarantee the security of the personal data processing. While Wuyts considers confidentiality as a security property, she highlights also its importance for preserving privacy properties, such as anonymity and unlinkability [21].

Similarly to linkability, information disclosure will also trigger all personal data processing related principles, since the data could be further collected, stored by third parties without specific purpose and without informing the DS. Thus, data minimisation and storage limitation principles cannot be complied with either. In addition, the accuracy of the personal data can be also jeopardized.

3.4.7 Unawareness

Unawareness occurs when a user is unaware of the information she is supplying to the system, and the consequences of her acts of sharing. In the era of digitalisation users tend to provide excessive information resulting in a loss of control of their personal information. Thus, awareness aims at ensuring that users are aware of their personal data and that only the minimum necessary information should be collected [21].

Unawareness points out to the violation of fairness and transparency requirements, since the DS is not informed of all the risks related to the personal data processing and was not provided all the information required in relation to their personal data processing. Unawareness also leads to the fact that the DS provides more personal information than required, and thus, the principles of data minimisation and purpose limitation are violated [21].

3.4.8 Non-compliance

Non-compliance is related to legislation, policy and consent and implies that the DS should be informed by the controller about the system's privacy policy and allows the DS to specify consent [21]. Wuyts gives some examples of non-compliance, such as incorrect privacy policies provided to the user or when the policy rules are incorrectly managed by the system administrator [21].

Wuyts notes that policy specifies one or more rules with respect to data protection and these are general rules determined by the stakeholders of the system; consent specifies one or more data protection rules and is determined by the user and only relate to the data regarding this specific user [21]. From a legal perspective, while the processing of personal data can be based on DS's consent, lawfulness of the processing is not limited to consent as a lawful ground for data processing activities. The GDPR provides for 5 additional legal grounds where the processing of personal data is not based on consent (Art. 6 GDPR).

When it comes to policy, Wuyts mentions compliance with internal policies of the company. However, compliance with internal policies of the company will not be enough if those policies are not correct, lack detail or are not user friendly with regard to privacy notices provided. Thus, non-compliance with policies should be related to broader issues covering also some external requirements and legal framework applying to controllers.

Non-compliance threat, as described in LINDDUN, seems to be too generic and lacks in precision. Its current wording suggests that all the data protection related legal frameworks will be triggered. However, eliminating this threat is easier said than done, since the legal compliance is like shooting at a moving target, as it continuously changes while you are working on it.

3.5 Privacy-Related Risk Knowledge Base

In this section we briefly describe the knowledge base created in ENACT and PDP4E projects. This knowledge base is the baseline for the recommendations provided in the risk management enabler, as well as the main database for the Automated Vulnerability Detector described in Section 3.2.

The proposed knowledge base is founded on the LINDDUN methodology to frame privacy-related threats, as well as on STRIDE to cover security issues. We use the threat trees proposed in LINDDUN as the starting point to populate our knowledge base. We connect the content of our knowledge base to public content in the Common Weaknesses Enumeration (CWE) list. Besides, we also leverage information from the Common Attack Pattern Enumeration and Classification (CAPEC) dictionary, through their link from CWE. In addition, our knowledge base contains information to facilitate the link of its content to known fines and penalties which data protection authorities within the EU have imposed under the GDPR.

3.5.1 Definition of Concepts to be Stored in the Knowledge Base

Figure 3.1 presents the class diagram published in [12] to model risk management concepts to allow to control privacy risks using DFDs. In particular, we consider each element of a DFD (Entity, Data Flow, Data Store and Process) a specific asset, according to LINDDUN methodology. This diagram is inspired by the UML diagram presented by Gupta *et al.* [8].

Based on this class diagram, our knowledge base will contain the following information:

- LINDDUN threat category: Our knowledge base will classify all vulnerabilities depending on one of the LINDDUN threat categories, namely,

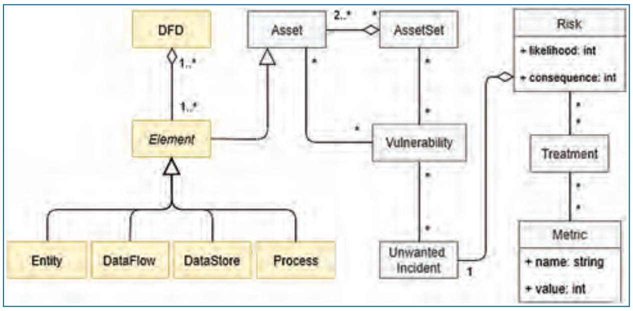


Figure 3.1. Class diagram to model risk management concepts using DFDs to describe system functionality [12].

Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness and Non-compliance. Note that LINDDUN is remarked because of the focus on this chapter on security issues. However, the knowledge base also uses STRIDE threats to classify privacy and security issues.

- Type of DFD component: we allow to express vulnerabilities of each of the four different types of components considered in a DFD, namely, entity, data flow, data store and process.
- Vulnerability information: We store information about vulnerabilities. This will include information such as a unique identifier, a short title, a longer description, etc. As mentioned before we will also need to store the conditions that are considered for a particular vulnerability to be relevant.
- Threat information: In order to simplify the information, we do not explicitly store the information about unwanted incidents, but we directly store the potential threats that may exploit a particular vulnerability.
- Control/Treatment information: we create a collection of controls that can be used as potential treatments to mitigate risks related to specific vulnerabilities and threats.

Based on this, in Figure 3.2, we describe the schema that represents the data schema of the information stored in this knowledge base.

The central concept of the schema is the *Vulnerability*. Vulnerabilities are connected to threats and threats are connected to controls. These link between these three types of elements constitutes the backbone of any risk management tool to provide options related to vulnerabilities and related threats and controls or mitigation actions. As explained in [11], vulnerabilities are also related to conditions. Besides, our knowledge base allows to establish dependencies among conditions. A certain condition may only make sense if another condition holds. For instance, let us assume a condition with id $c1$ that evaluates the following question: “Does

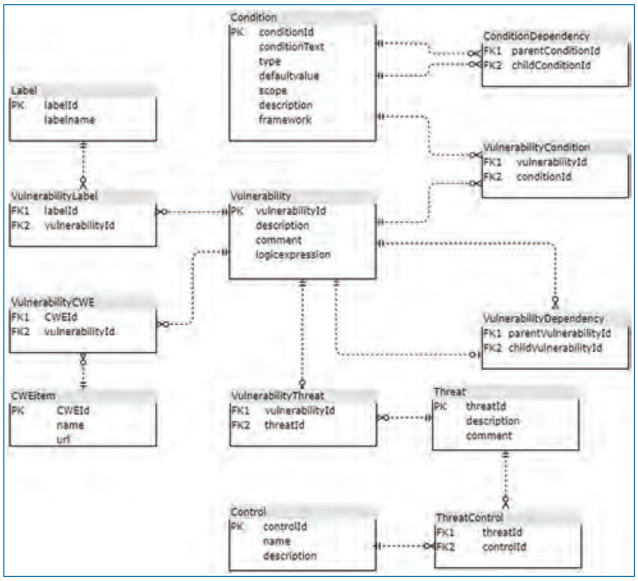


Figure 3.2. Knowledge base schema.

this entity represent a DS or a proxy to DSs?" From a privacy perspective, if $c1$ is true then other conditions may be relevant such as for instance "Does this entity need to be authenticated to execute this DFD?" or "Are credentials of this entity shared with any potentially untrustworthy receiver entity?". So, the relevance of conditions may depend on the evaluation of other conditions.

Besides, vulnerabilities are related to other vulnerabilities. In reality, inspired by STRIDE and LINDDUN, we have also created and extended threat trees in the knowledge base. Therefore, a vulnerability may be decomposed in more detailed vulnerabilities in a structure shaped like a tree. Relation *VulnerabilityDependency* represents the link between vulnerabilities in the tree. *Vulnerability* relation is also related to *Label* relation and *CWEitem* relation. The first link represents keywords related to the vulnerabilities that we will need later on to look for cases in the GDPR Enforcement Tracker.[4] This way, we will be able to show actual cases of GDPR enforcement in cases related to this type of vulnerability. For instance, "Information disclosure of a process" may be labelled with "Confidentiality" label or vulnerability named "Data is not encrypted" may be labelled as "Encryption". The actual mapping between cases and labels has been implemented in the PDP4E project. The second link corresponds to a manual exercise to connect the STRIDE vulnerabilities with CWE items. When the link is established, our enabler uses the url stored

4. www.enforcementtracker.com

in the database for each CWE item and scans online information to automatically detect mitigation actions and add them as controls in the *Control* relation. We also follow links to the CAPEC database related to attack patterns related with that vulnerability and store the information in the *Threat* relation and potential connected mitigation actions in the Control relation.

3.6 IoT Use Cases Description

Risk Management enables building trustworthy systems. Especially when considering SIS because of the way multiple independent devices connect together and exchange data. To illustrate this, the following section will describe the kind of challenges found in an IoT use case in terms of privacy risks for a DS. Privacy is specially challenging in IoT systems, not only because IoT systems rely on technology that is still not mature and in continuous evolution, but also because the higher integration of devices in the physical world makes those devices actual proxies to DS. For instance, the location of a device may easily act as a mechanism to deduce the location of a DS. For this purpose, following we describe a use case which is particularly relevant under this point of view.

3.6.1 Connected Vehicles

In this section we present an IoT-related use case focused on communications between vehicles and other vehicles or the road infrastructure, also called V2X. More specifically, the use case focused on the Cooperative Awareness Messages protocol [6], a part of the Cooperative Intelligent Transportation System (C-ITS) ecosystem which aims to make the different actors of road infrastructure share information on vehicle status, traffic, road works, etc... This protocol specifies how vehicles can share data about their position and status, like speed or heading, with other vehicles around them. This type of communication is of particular interest for the development of advanced driver-assistance systems (ADAS) as well as autonomous vehicle. It allows a vehicle to build a map of its surrounding and track nearby vehicles to anticipate possible incidents.

We chose to consider this system because it highlights a potential loss of control of data related to a DS, which can lead to significant privacy issues. The V2X network on which the CAM messages are transmitted is a local radio network. Because this system aims to inform other participants in the neighborhood about the status of the vehicle, messages are broadcast to every station capable of listening to the network. A consequence of broadcasting is that these messages cannot be encrypted with anything else than a key shared among all participants. As a result, the data they carry can effectively be read by any station with access to the network without control of the sender.

Because the data received from these messages can be used to trigger warnings to drivers or even collision avoidance systems, it is necessary to ensure that they come from a reliable source. One of the mechanisms deployed to this end is the use of cryptographic signatures to authenticate the messages. When signing a message, a vehicle uses a pair of cryptographic keys provided by an external authority trusted by every participant of the network. This signature shows that they have been authorized to send messages and the receivers will be able to trust the data they send.

If used in a privately-owned vehicle, this system can raise issues about the privacy of the owner. First, to provide the information about the vehicle, a lot of parameters are collected by the equipment which generates those messages. Those parameters show when and where the vehicle has been driven, but also give information about the behaviour of the driver, like the speed of the vehicle. If these parameters were recorded and stored for analysis, they could be used to determine the driving habits of the owner. On top of that, because of the signature they carry, it can be possible to classify the messages by which vehicle has sent them. This could make mass recording of messages a source of personal data leakage if a link between a signature and specific vehicles can be made.

Relevant attack scenarios

The CAM system could be used in different ways which have consequences on the privacy of the owner of the vehicle. For example, it could be used to tail a specific vehicle, or the data collected inside the transmission equipment could be harvested by a malicious entity. For the purpose of this demonstration, we will use an attack scenario that leverages the signatures used to authenticate vehicles to track the behaviour of a specific vehicle. This scenario could be carried out by recording messages thanks to stations deployed across the road infrastructure. Because the vehicle is registered to a specific owner, such a record would allow to get the trip history of the owner and determine his usual destinations. The time at which the messages are emitted can also be used to build his schedule, the additional information on vehicle status like speed, light status, brake indicator can be used to analyse his driving profile. This data could be, for example, used by insurance companies to determine fees, or to show that an individual goes regularly to a political meeting.

3.6.2 Practical Implementation Aspects

All the contributions presented in this chapter have been implemented in the latest version of the ENACT Risk Management enabler in collaboration with the

PDP4E project. In particular, the enabler includes the extended version of the OWASP Risk Rating methodology presented in Section 3.3. Specifically, the questions suggested for impact analysis have been added to the enabler. Besides, the mapping between LINDDUN threat categories and GDPR data processing principles and DS rights have been embedded in the enabler. The enabler has a GDPR-based dashboard that allows to understand the overall level of risk for each principle and DS rights, based on the status of each risk detected in the enabler. Finally, the knowledge base described in 3.5 has been also embedded in the enabler, this enables not only the automated detection of vulnerabilities, but also benefiting from open databases such as CWE or CAPEC.

3.7 Risk Management Enabler Evaluation

This section aims to demonstrate the usage of the enabler on the connected vehicle use case. The enabler is evaluated based on its ability to help the user identify privacy risks in the use case and suggest meaningful treatments to manage them. The results are also compared with the way these issues are currently handled to demonstrate how adequate the enabler with respect to real world constraints.

Figure 3.3 presents the Data Flow Diagram used to support the attack scenario described in the previous section. It features two main processes: Key Provisioning and C-ITS App. The Key provisioning process is in charge of generating the key used by the vehicle to sign the messages it will send. The C-ITS App is collecting information on the vehicle position and status from the GPS and sensors to generate the messages, sign them and send them to the CAM network.

In the attack scenario, we consider the possibility of recording messages sent to the network. This corresponds to the data flow sending signed messages between the C-ITS process and the CAM Network entity.

The Risk Management enabler allows providing information on the role of these elements through the questionnaire described in [11] and used to fit the Automated Vulnerability Detector (AVD), presented in Section 3.2. This questionnaire contains the conditions used to discriminate the relevance of potential vulnerabilities and it is stored in the knowledge base, as described in Subsection 3.5. Table 3.1 gives some examples of questions that the user has to answer.

In particular, the question related to anonymous communication triggers several vulnerabilities based on the threat trees defined by LINDDUN. We now analyze a particular example based on the linkability of a data flow threat tree defined, presented in Figure 3.4. Those vulnerabilities are presented in Table 3.2. The vulnerabilities on linkability based on computer or session ID are both similar to the potential privacy issue of using the vehicle signature to track it. In this case, the

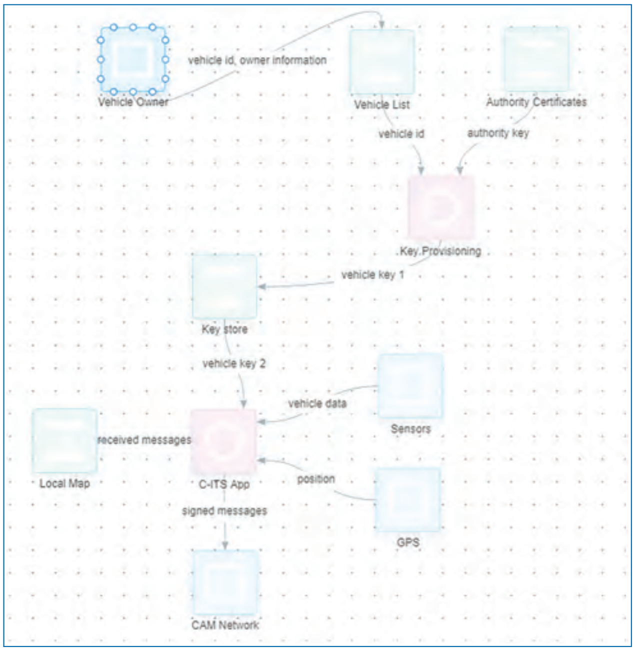


Figure 3.3. Data flow diagram of the C-ITS use case.

Table 3.1. Sample questions.

Type of Element	Question	Answer
Data Flow	Is the communication channel wireless?	Yes
	Are messages sent through this data flow encrypted?	No
	Is anonymous communication used?	No
Entity	Does this entity represent a DS or a proxy to DSs?	No
	Could this entity be or become untrustworthy (now or in the future)?	Yes

public key used for the signature serves as the ID linking together different messages, posing a privacy threat as an attacker may be able to continuously monitor a vehicle, and using other means to finally identify the driver, learn detailed information about her location, movements, habits, etc…

From a pure security point of view these privacy vulnerabilities would probably not be highlighted. On the contrary, they are probably desirable in some way as being able to trace the origin of a communication helps avoiding security issues related to repudiation or authentication, which is the reason why this signature has been added to the messages.

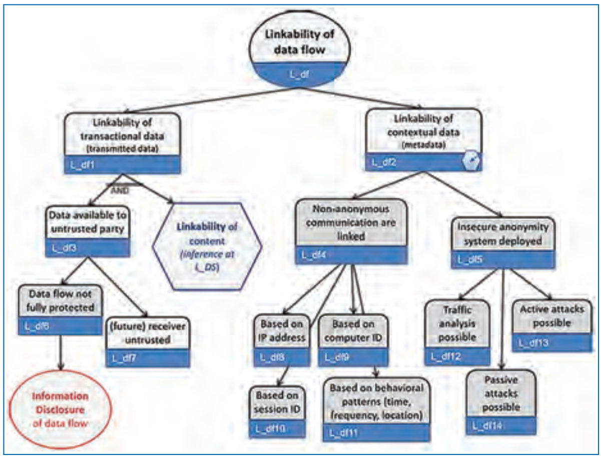

Linkability of data flow
L_df
Linkability of transactional data (transmitted data)
L_df1
AND
Linkability of contextual data (metadata)
L_df2
Data available to untrusted party
L_df3
Linkability of content (inference at L_DS)
Non-anonymous communication are linked
L_df4
Insecure anonymity system deployed
L_df5
Data flow not fully protected
L_df6
(future) receiver untrusted
L_df7
Based on IP address
L_df8
Based on computer ID
L_df9
Traffic analysis possible
L_df12
Active attacks possible
L_df13
Information Disclosure of data flow
Based on session ID
L_df10
Based on behavioral patterns (time, frequency, location)
L_df11
Passive attacks possible
L_df14

Figure 3.4. LINDDUN threat tree for linkability in data flows from https://www.linddun.org/linkability

Table 3.2. Privacy vulnerabilities detected.

Vulnerability	LINDDUN Property	Category	Associated Risk
Based on computer ID	Linkability	Non-anonymous Communication traced to entity	The data flow can be linked on computer ID and an attacker could link the ID to a person.
Based on session ID	Linkability	Non-anonymous Communication traced to entity	The data flow can be linked on session ID and an attacker could link the ID to a person.
Based on behavioural patterns	Linkability	Non-anonymous Communication traced to entity	Packet Counting Attacks, Timing attacks

If we consider more specifically the risk where data flows share a computer ID that can be linked to a person, the following controls are proposed by the Risk Management enabler:

- **Use tunneling through a Virtual Private Network (VPN).** This control aims at hiding the actual sender by using the exit point of the VPN as a public address. The messages can still be linked together but linking to the sender is harder.
- **Randomizing the computer ID.** This control uses unpredictable identifiers to break the link between messages. It may still be possible to link an identifier to a person but only the messages sent with this identifier will be impacted.
- **Use temporary identifiers which can be reused by different individuals across different periods of time.** This control hides the origin of messages

among the different participants. Both identifying the sender and linking the messages together is harder.

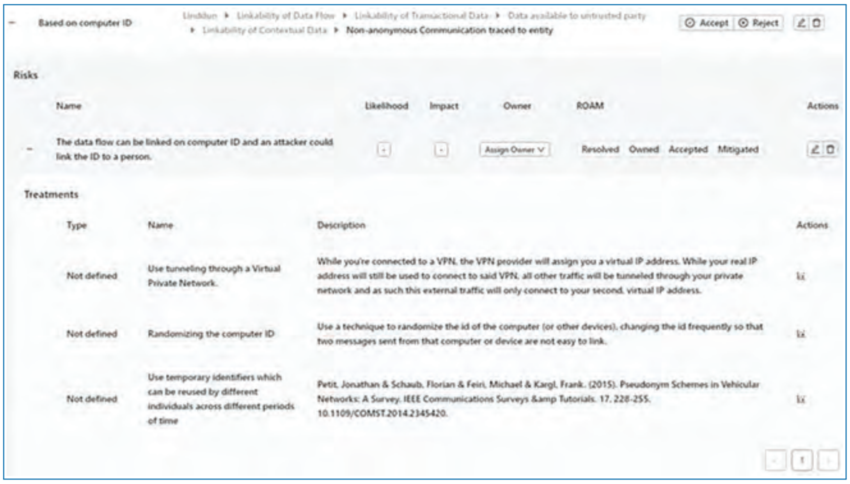
Out of these controls, the solution chosen to address this linkability issue in the described C-ITS use case is to randomly choose identifiers, called pseudonyms, in a pool assigned to a vehicle. This approach is also described in [7] and corresponds to the "Randomizing the computer ID" control described before as it makes it more difficult to link different messages by introducing randomness when choosing the pseudonym. This results in a more complex overall system as these identifiers will also need to be certified by a trusted authority but achieves a compromise between the security and privacy requirements. As an example, the ETSI Technical Report[7] also mentions the possibility of exchanging pseudonyms between vehicles. However, because multiple senders could be associated with the same pseudonym, this exchange would prevent law enforcement as access to the identity of a sender would not be available when required in an investigation. As the pseudonym scheme still allows for limited disclosure of identity, the impact on DSs is kept to a minimum while still addressing security requirements.

With this example, we demonstrated a full iteration of the risk management cycle, going from a guided identification of vulnerabilities to the selection of treatments. As shown the privacy risks identified and treatments selected are consistent with real privacy issues identified and discussed in the context of C-ITS systems. The knowledge base, being based on CWE and CAPEC, is definitely more suited for more classic IT systems than C-ITS systems in the way it describes issues and treatments so issues more specific to the domain considered are likely to be missed. For example being able to send forged messages could lead to traffic disruption. However, despite these shortcomings, it is still able to manage fairly specific issues.

The screenshot from Figure 3.5 shows how the enabler helps its user by offering direct access to potential risks and controls from the selection of vulnerabilities. This presentation allows to have a better view of potential consequences of a vulnerabilities, it also saves the engineer's time by grouping together the relevant information about a vulnerability and its associated risks.

3.7.1 Analysis of the Extended OWASP Risk Rating Methodology

In this section, we will evaluate how the modified OWASP risk appraisal method described before can help having a better view of the impact of a risk on privacy. This analysis will focus on the impact evaluation part because likelihood evaluation has not been modified.


Based on computer ID
LINDDUN ▸ Linkability of Data Flow ▸ Linkability of Transactional Data ▸ Data available to untrusted party ▸ Linkability of Contextual Data ▸ Non-anonymous Communication traced to entity
Accept
Reject

Risks

Name	Likelihood	Impact	Owner	ROAM	Actions
The data flow can be linked on computer ID and an attacker could link the ID to a person.	-	-	Assign Owner	Resolved Owned Accepted Mitigated	

Treatments

Type	Name	Description	Actions
Not defined	Use tunneling through a Virtual Private Network.	While you're connected to a VPN, the VPN provider will assign you a virtual IP address. While your real IP address will still be used to connect to said VPN, all other traffic will be tunneled through your private network and as such this external traffic will only connect to your second, virtual IP address.	
Not defined	Randomizing the computer ID	Use a technique to randomize the id of the computer (or other devices), changing the id frequently so that two messages sent from that computer or device are not easy to link.	
Not defined	Use temporary identifiers which can be reused by different individuals across different periods of time	Petit, Jonathan & Schaub, Florian & Feiri, Michael & Kargl, Frank. (2015). Pseudonym Schemes in Vehicular Networks: A Survey. IEEE Communications Surveys &amp Tutorials. 17. 228-255. 10.1109/COMST.2014.2345420.	

Figure 3.5. Screenshot from the risk management enabler showing the presentation of risks and controls.

Taking the example of the Identification based on machine ID risks that was considered before, with the classic OWASP methodology, the impact would be evaluated as presented in Table 3.3. This results in a technical impact of 3 and a business impact of 4.75. We choose the maximum of both categories as the final impact which gives a 4.75 impact score. This corresponds to a Medium rating according to the scales defined in the OWASP methodology.

Following, Table 3.4 presents the evaluation of the new categories introduced to measure impact on the DS. By reorganizing the previous score to take into account that the *Privacy violation* factor is now in the Privacy impact category and it is renamed as *Scale* according to Section 3.3, we get a technical impact of 3, business impact of 3.33, and privacy impact of 7. The new impact score is 7 which is a High rating.

As expected, the privacy-oriented factors indicate a significant potential impact for the privacy of a person using this system. Considering each impact category separately and using the score of the most important one as the final impact avoids minimising the contribution of a category if the others are rated low. This is an important aspect as security and privacy can be at odds with each other and the apparition of compensation mechanisms where one low-rated aspect could reduce the overall impact and undermine the proper assessment of risks.

3.7.2 Connecting the use Case with GDPR

Besides, the Risk Management enabler will establish a link with respect to GDPR principles and DS rights. This is an important benefit as, to our knowledge, it is

Table 3.3. OWASP evaluation of the identification by machine ID risk.

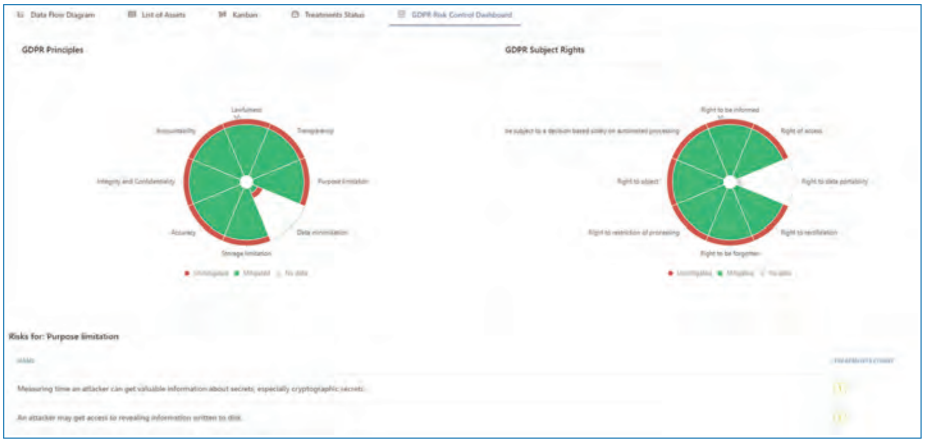
Factor	Value	Justification
Technical Impact		
Loss of confidentiality	1	The data sent through this data flow is not encrypted therefore it is not confidential
Loss of integrity	1	The integrity of the data is not affected by this risk
Loss of availability	1	The availability of either this communication or the CAM service is affected by this risk
Loss of accountability	9	The threat agent can listen passively to the network
Business Impact		
Financial damage	1	No financial damage will come directly to the maker of the system by exploiting the risk.
Reputation damage	5	Exploiting the risk can lead customers to be suspicious of using the system
Non-compliance	4	The system does not take into account privacy concerns like anonymity so it does not comply with GDPR, but it complies with technical specifications which also do not take into account privacy.
Privacy violation	9	The number of people affected can be measured in the millions

Table 3.4. Extended OWASP impact evaluation of the identification by machine ID risk.

Factor	Value	Justification
Harm	7	The information collected can be used to track the movements of a person. It can also be used to profile its driving style which could be used by insurance companies.
Sensitivity	5	Knowing where a person went can help deduce a political or religious affiliation
Expectation	7	The expectation on the CAM system is that messages are only processed locally by the surrounding vehicles and is not disseminated beyond the neighbourhood

the first tool that allows to map technical risks and controls with GDPR concepts, which are established from a legal perspective.

In the particular IoT use case presented above in this section, linkability issues have been detected related to the particular attack scenario under analysis. However, it is not straightforward to understand and explain, in front of a potential



Risks for: Purpose limitation

Measuring time an attacker can get valuable information about secrets, especially cryptographic secrets.

An attacker may get access to revealing information written to disk.

Figure 3.6. Example of dashboard the risk status categorized by GDPR principles and DS rights.

explicit audit, what are the practical steps that we are considering from a technical perspective, to protect DS rights and GDPR principles, a part from a vague understanding that the mentioned threat may affect confidentiality.

Once the risks are detected and the mitigation actions selected, we can mark risks as mitigated if their severity was considered high enough and controls are deemed sufficient to accept the residual risk.

Thanks to the analysis performed in Section 3.4, we are able to connect linkability threats to different related GDPR principles such as lawfulness, transparency, purpose limitation, data minimisation, storage limitation, accuracy, integrity and confidentiality or accountability. Also, they can be connected to DS rights such as rights to be informed, of access, to data portability, to rectification, to be forgotten, to restriction of processing, to object and not to be subject to a decision based solely on automated processing. Omitting the responsibility for adequately managing these privacy-related risks, as the IoT system architect and developer, goes against GDPR and may involve harm to users and other DSs as well as penalties.

The Risk Management enabler will help users to control the impact from GDPR perspective, showing a GDPR-specific dashboard that may specify the level of mitigation of risks related to each GDPR principle and DS right. An example is depicted in Figure 3.6.

3.8 Conclusions and Future Work

In many cases, security and privacy are conflicting requirements. While security has been largely explored for decades, privacy is a much more immature area. This is specially true when we try to control privacy in digital ecosystems based on newer and evolving technologies such as in the case of IoT systems.

In this chapter we have contributed to eliminate different roadblocks to achieve a better control of privacy through risk management. One of these is the capacity to improve risk assessment under the scope of privacy, essential for IoT systems, but also in general for any type of digital system. A second one is the capacity to connect the management of technical risks controlled by architects, developers and risks analysts based on privacy-related threat analysis methodologies like LINDDUN, with current legal frameworks to protect privacy such as GDPR. In particular, the level of connectivity between the GDPR concepts and LINDDUN threat categories is very large since they rely on different vocabulary and it is difficult to establish a 1:1 relationship between concepts. This interdisciplinary exercise is one of the first attempts to bridge the existing gap between the legal approach towards privacy risks and engineers approach towards privacy risks.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No. 780351 and No. 787034.

References

[1] Atif Ahmad, Justin Hadgkiss, and Anthonie B Ruighaver. "Incident response teams–challenges in supporting the organisational security function". *Computers & Security* 31.5 (2012), pp. 643–652.

[2] Barry Boehm and Richard Turner. *Balancing agility and discipline: A guide for the perplexed, portable documents*. Addison-Wesley Professional, 2003.

[3] Sean Brooks *et al. An introduction to privacy engineering and risk management in federal systems*. US Department of Commerce, National Institute of Standards and Technology, 2017.

[4] Tom DeMarco. "Structure analysis and system specification". In: *Pioneers and Their Contributions to Software Engineering*. Springer, 1979, pp. 255–288.

[5] Mina Deng Deng *et al.* "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements". In: 16 (2011), pp. 3–32.

[6] *ETSI EN 302 637-2 V1. 3.1-Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative*

Awareness Basic Service, 2014. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.01_30/en_30263702v010301v.pdf.

[7] *ETSI TR 103 415 V1.1.1-Intelligent Transport Systems (ITS); Security; Pre-standardization study on pseudonym change management*, 2018. URL: https://www.etsi.org/deliver/etsi_tr/103400_103499/103415/01.01.01_60/tr_103415v010101p. pdf.

[8] Smrati Gupta *et al.* "Risk-driven framework for decision support in cloud service selection". In: *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE. 2015, pp. 545–554.

[9] Mireille Hildebrandt. *Smart technologies and the end (s) of law: novel entanglements of law and technology*. Edward Elgar Publishing, 2015.

[10] Susan Eva Landau. *Listening in: Cybersecurity in an insecure age*. Yale University Press, 2017.

[11] Victor Muntés-Mulero *et al.* "Enabling Continuous Privacy Risk Management in IoT Systems". In: *Security Risk Management for the Internet of Things: Technologies and Techniques for IoT Security, Privacy and Data Protection*. Edited by John Soldatos. Now Publishers, 2020.

[12] Victor Muntés-Mulero *et al.* "Model-driven Evidence-based Privacy Risk Control in Trustworthy Smart IoT Systems". In: (2019).

[13] Data Protection Working Party. *Guidelines on Data Protection Impact Assessment (DPIA) and determining whether processing is "likely to result in a high risk" for the purposes of Regulation 2016/679*, 2017.

[14] The Article 29 Working Party. "Opinion 4/2007 on the concept of personal data". 01248/07/EN WP 136.

[15] Andreas Pfitzmann and Marit Hansen. "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management". In: (2010).

[16] Adam Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[17] Daniel J Solove. "A taxonomy of privacy". In: *U. Pa. L. Rev.* 154 (2005), p. 477.

[18] Elizabeth Stoycheff *et al.* "Privacy and the Panopticon: Online mass surveillance's deterrence and chilling effects". In: *New media & society* 21.3 (2019), pp. 602–619.

[19] Isabel Wagner and Eerke Boiten. "Privacy risk assessment: from art to science, by metrics". In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2018, pp. 225–241.

[20] Jeff Williams. "Owasp risk rating methodology". In: *Library Catalog: owasp.org. url: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology (besucht am 05. 06. 2020)*, (2020).

[21] Kim Wuyts. “Privacy Threats in Software Architectures”. In: (2015).

[22] Zheng Yan, Peng Zhang, and Athanasios V. Vasilakos. “A survey on trust management for Internet of Things”. In: *Journal of network and computer applications*, 42, (2014), pp. 120–134.

DOI: 10.1561/9781680838251.ch4

Chapter 4

Model-based Continuous Deployment of SIS

By Nicolas Ferry, Hui Song, Rustem Dautov, Phu Nguyen and Franck Chauvel

4.1 Introduction

Smart IoT Systems (SIS) are characterized by the presence of software deployed along the entire IoT-Edge-Cloud continuum. Software defines the behaviour of the SIS, and such behavior keeps evolving during the entire system life cycle following the ever-changing system context. This evolution may be realized as self-adaptation (such as the use of online learning for dynamic adaptation, as elaborated in Chapter 6) or manual reconfiguration of the existing software, but, very often, it requires releasing new versions of software. On the one hand, this evolution characteristic of SIS is typically conflicting with the traditional IoT systems vision consisting of devices with immutable code, once deployed at the factories. There must be new approaches for supporting the evolution of SIS. DevOps, on the other hand, promotes the idea of continuously delivering new software updates. Indeed, the DevOps movement promotes an iterative and incremental approach enabling the continuous evolution of software systems. Embracing DevOps can support the continuous evolution of SIS and improve their trustworthiness (e.g., security). As an evolution of the DevOps movement, DevSecOps [30] promotes security as an aspect that must be carefully considered in all the development and operation phases for the continuous evolution of systems to be secure. However, how to effectively deploy the software update to the computing continuum is a main obstacle to enable DevOps or DevSecOps for SIS as it requires the capability of continuous deployment of software at all levels.

Continuous and automatic software deployment is still an open question for SIS, especially at the Edge and IoT ends. The state-of-the-art Infrastructure as Code (IaC) solutions are established on a clear specification about which part of the software goes to which types of resources. This is based on the assumption that in the Cloud it is easy to obtain the exact computing resources as required. However, this assumption is not valid on the Edge and IoT levels. A typical SIS in production often contains hundreds or thousands of heterogeneous and distributed devices (also known as a *fleet of IoT/Edge devices*[1]), each of which has a unique context, while their connectivity and quality are not always guaranteed. The major challenges are as follows:

- How to automate the deployment of software on heterogeneous devices possibly with limited or no direct Internet access?
- How to manage variants of the software which fit different types or contexts of Edge or IoT devices in the fleet?
- How to ensure the trustworthiness of the deployed software whilst the quality of the underlying resources are not guaranteed?

In the ENACT project, we focus on the problem of automatic software development for SIS, and our research attempts to address these challenges resulted in two complementary prototype tools for the deployment of SIS at two different layers: (i) GeneSIS targets at the device layer, providing a unified way to deploy software on heterogeneous devices, including those without direct internet connection; (ii) DivEnact targets at the fleet layer, allowing developers to deploy software into the abstract fleet as a whole instead of focusing on concrete individual devices. The tool maintains the software variants and assigns them automatically to the devices according to their contexts. With trustworthiness (e.g., security) being a concern cross-cutting both layers, our tools provide solutions that contribute making the deployment and the SIS trustworthy. At the device layer, we support the specification and deployment of security and privacy mechanisms together with the SIS software in a DevSecOps fashion. Moreover, we provide the novel *rolling deployment* method to guarantee the availability of the deployed software as well as to handle errors during a deployment, i.e., in addition to the main software, we also deploy a

1. Similar to a fleet of vehicles in a transportation company, a fleet of devices are owned by the same application providers and distributed to different places or users. Devices in a fleet conduct relatively independent tasks, whilst coordinated by the application provider from a global perspective.

backup copy which will replace the main one when necessary without delay. At the fleet level, we maintain the software diversity within the fleet for security purposes.

Model-Driven Engineering (MDE) is the scientific basis underlying both GeneSIS and DivEnact. MDE is a branch of software engineering that aims at improving the productivity and cost-effectiveness of software development by shifting the paradigm from code-centric to model-centric. It has shown to be effective in supporting design activities [42]. This approach, which is commonly summarised as "model once, generate anywhere", is particularly relevant to tame the complexity of developing heterogeneous systems such as SIS. Models and modelling languages as the main artefacts of the development process enable developers to work at a high level of abstraction by focusing on deployment concerns rather than implementation details.

This chapter is organized as follows. Section 4.2 provides an overview of the current state of the art and of the practice for the automatic deployment of SIS. Section 4.3 introduces our solutions for the automatic deployment of SIS, first describing how they can be integrated in order to form a coherent deployment bundle and then detailing each our two enablers: GeneSIS and DivENACT. Section 4.4 focus on the support offered by our solutions to ensure the trustworthiness deployment of SIS. Finally, Section 4.5 draws some conclusions.

4.2 The State of the Art

Software deployment has been evolving from deployment of component-based commercial desktop software [39], deployment of component-based distributed applications [25], to deployment on Cloud resources, and more recently deployment for IoT systems along the entire IoT-Edge-Cloud continuum. Even though some core concepts from deployment of component-based applications such as *capability*, *port* in [25] can be inherited for deployment on Cloud or IoT resources, they need to be tailored and customized to fully address the specificities of these environments.

4.2.1 On the Deployment at the Device Layer

For some years now, multiple tools have been available on the market to support the deployment and configuration of software systems, *e.g.*, Puppet,[2] Chef.[3] These tools were first defined as configuration management tools aiming at automating

2. https://puppet.com/

3. https://www.chef.io/chef/

the installation and configuration of software systems on traditional IT infrastructure. Recently, they have been extended to offer specific support for deployment on Cloud resources. Meanwhile, new tools emerged and were designed for deployment of Cloud-based systems or even multi-Cloud systems (*i.e.*, systems deployed across multiple Clouds from different providers) such as CloudMF [19], OpenTOSCA [43], Cloudify,[4] and Brooklyn.[5] Those are tailored to provision and manage virtual machines or PaaS solutions. In addition, similar tools focus on the management and orchestration of containers, *e.g.*, Docker Compose,[6] Kubernetes.[7] As opposed to hypervisor virtual machines, containers leverage lightweight virtualization technology, which executes directly on the operating system of the host. As a result, the container engine shares and exploits a lot of resources offered by the operating system thus reducing containers' footprint. These characteristics make container technologies suitable not only for the Cloud, but also for Edge devices [13].

Besides, a few tools, such as Resin.io (Balena)[8] and ioFog,[9] are specifically designed for the IoT. In particular, Resin.io provides mechanisms for (i) the automated deployment of code on devices, (ii) the management of a fleet of devices, and (iii) the monitoring of the status of these devices. Resin.io supports the following continuous deployment process. Once the code of the software component is pushed to the Git server of the Resin.io Cloud, it is built in an environment that matches the targeted hosting device(s) (*e.g.*, ARM for a Raspberry Pi) and a Docker image is created before being deployed on the hosting device(s). However, Resin.io offers limited support for the deployment and management of software components on tiny devices that cannot host containers.

Regarding the deployment of elements of hardware and software that are to operate in harmony within a networked system, the Software Communications Architecture (SCA) [1] and IoT deployment share some basic concepts. The SCA is an open architecture that specifies a standardized infrastructure for a software-defined radio (SDR). However, the SDR SCA specification requires an SCA-compliant system for elements of hardware and software to operate within. In other words, the SCA is tightly tied to the specific needs for standardizing the development of SDRs, which is much less heterogeneous than the IoT domain in terms of

4. http://cloudify.co/
5. https://brooklyn.apache.org
6. https://docs.docker.com/compose/
7. https://kubernetes.io
8. https://www.balena.io/
9. https://iofog.org/

communication means, systems of systems, which may span all the layers of Cloud, Edge, IoT devices. Moreover, the SCA does not have any concept about supporting the deployment on devices not directly accessible.

In [34], we conducted a systematic literature review (SLR) to systematically study a set of 17 primary studies of orchestration and deployment specifically for the IoT. We found a sharp increase in the number of primary studies published in two-three recent years. We also found that most approaches do not really support the IoT deployment and orchestration at low-level IoT devices. As for the continuous deployment tools mentioned before, these approaches mainly focus on the deployment of software systems over edge and Cloud infrastructures whilst little support is offered for the IoT space. When this feature is available, it is often assumed that a specific bootstrap is installed and running on the IoT device. A bootstrap is a basic executable program on a device, or a run-time environment, which the system in charge of the deployment rely on (*e.g.*, Docker engine). Approaches such as Calvin run-time [28], WComp [27], or D-LITE [10], D-NR [24] all rely on their specific run-time environment where mechanisms such as dynamic component loading or class loading are typically used. There is a lack of addressing the trustworthy aspects and advanced support in the deployment and orchestration of the IoT.

To the best of our knowledge, none of the approaches and tools aforementioned have specifically been designed for supporting deployment over the whole IoT, Edge, and Cloud infrastructure. In particular, they do not provide support for deploying software components on IoT devices with no direct or limited access to internet. In addition, we also identified they do not offer support for including security concerns as core concepts in the tool and/or language.

4.2.2 On the Deployment at the Fleet Layer

While all these solutions discussed above are focusing on the deployment of a software system, they typically do not offer specific support for the management of a fleet of devices or a fleet of systems, which basically consists in managing large set of deployments with those solutions.

To the best of our knowledge, there is no effective solution to this *fleet deployment* problem. The start-of-the-art Infrastructure as Code (IaC) tools automate the deployment of one application on one device, or a predefined set of devices, but lack the support for distributing multiple variants across a large fleet. They also do not provide sufficient automated support for updating devices with constrained resources and limited (or none) Internet connectivity [34]. Such embedded and microcontroller-enabled devices traditionally have been flashed with 'one-off' firmware not intended to be updated in the future, but they are not often seen as active contributors to the common pool of shared computing resources, which can be iteratively assigned and deployed with updated firmware. This has also led to the so-called concept of IoT-edge-cloud computing continuum, where computing and storage tasks are distributed across all three levels. On the other hand, the mainstream IoT/Edge fleet management platforms offer tools to maintain multiple deployments, the fleet of devices, and their contexts, but developers still need to manually designate which deployment goes to which device.

Another relevant reference architecture for deploying component-based applications into heterogeneous distributed target systems is described in [37]. In particular, the proposed architecture includes the concept of *Planner* – a component responsible for matching software requirements to available platform resources and deciding whether a component is compatible with a device. These existing specifications remain implementation-agnostic and only describe the high-level concepts. Software diversity is a new dimension of architecture-level properties, which is both a result of the hardware heterogeneity and a method toward more secure system. The fleet deployment approach provides a implementation-level support to our theoretical approaches towards a more diverse software [29, 45].

The assignment problem (such as assigning software components to the devices in an IoT fleet) frequently appears in ICT scenarios, where some resources need to be allocated to available nodes, often taking into consideration various context-specific characteristics [38, 40]. The research community has come up with multiple algorithms, ranging in their computational complexity, completeness, preciseness, etc. Many of these approaches treat assignment as a collection of constraints, which need to be satisfied in order to find an optimal solution in the given circumstances [2, 7]. The approaches based on Satisfiability Modulo Theories (SMT) are specifically popular and efficient due to their expressively and rich modelling language [8]. In this respect, a relevant approach that also makes use of SMT and Z3 Solver is described by Pradhan *et al.* [41]. The authors introduce orchestration middleware, which continuously evaluates available resources on Edge nodes and re-deploys software accordingly. Similar goal is pursued by Vogler *et al.* in [46], where authors report on a workload balancer for distributing software components at the Edge. Multiple approaches specifically focus on the autonomic and wireless nature of IoT devices and contribute to energy-efficient resource allocation, where the primary criterion for software deployment is energy efficiency [47]. A main obstacle for using SMT in practice is the gap between real platforms and the mathematical model.

Model-based techniques are often used to support DevOps. Combemale *et al.* [12] present an approach to use a continuum of models from design to runtime to accelerate the DevOps processes in the context of cyber-physical systems. Artavc *et al.* [3] uses deployment models on multiple Cloud environments, which is a promising way to support the smooth transition of software from testing to production environments. Looking at approaches targeted at particular application domains, Bucchiarone *et al.* [9] use multi-level modelling to automate the deployment of gaming systems. In [16, 17], the authors apply model-driven design space exploration techniques to the automotive domain and demonstrate how different variants of embedded software are identified as more beneficial in different contexts, depending on the optimisation objective and subject to multiple constraints in place. To solve this optimisation problem, the authors also employ the SMT techniques and the Z3 solver implementation.

4.3 Overview of the ENACT Deployment Bundle

The ENACT approach to automatic software deployment is implemented as a prototype deployment bundle with two enablers, i.e., GeneSIS and DivEnact, supporting automatic deployment at the device and fleet layers, respectively.

Figure 4.1 illustrates how the ENACT deployment bundle is used in a typical SIS. The illustrative SIS has six subsystems, each of which is in charge of a particular business task, such as serving a user, monitoring a room, etc. Such a subsystem is usually composed by at least one edge device and several IoT devices such as sensors and actuators. For the sake of simplicity, we do not show all the IoT devices. These subsystems form the fleet of this SIS. Since each subsystem contains one main edge device as the main contact point, or gateway with the back-end service, we also refer to such fleet as an *edge fleet*. A fleet is normally distributed, with the edge devices (together with its IoT devices) serving different customers or tenants, and deployed in different locations. The developers often maintain one or several edge devices at their own premises for testing or trial purposes.

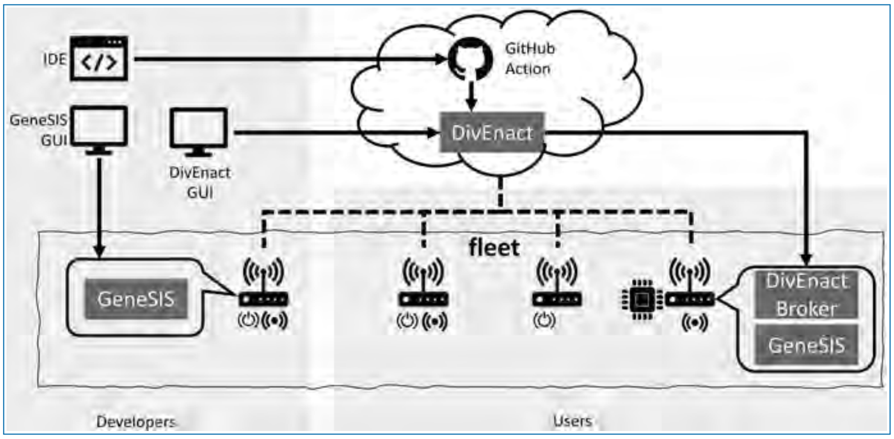


Figure 4.1. The ENACT deployment bundle.

GeneSIS supports the automatic deployment within a local subsystem, for example the deployment on the devices located on the developers' side. In such case, the developers can directly interact with the GeneSIS engine hosted on the local edge device, and use it as the bridge to further deploy required code to the associated IoT devices. In the development phase, developers define a *deployment model* in the GeneSIS modelling language specifying which software artefacts should be deployed onto which devices. Once the development phase completed, in the deployment phase, the same deployment model, or a slightly modified one, will be provided to the GeneSIS deployment engine, running either on a local machine or the edge device. The engine will install or update the software artefacts according to the deployment model.

DivEnact handles a different automatic deployment problem at the fleet level. When the developers want to release the new version of their application to production, they need to deploy software artefacts to all the devices on the users' sites. They cannot extend the deployment model to include every device in the fleet, because such a huge model is not maintainable, especially when the devices keep joining and exiting the fleet. Instead, since each user has a subsystem similar to the one at the developers' side, the developers can provide the deployment models they developed in the previous phase for the local subsystem to DivEnact. The latter maintains the list of all subsystems, and sends the deployment model to the devices before invoking the GeneSIS engine running on the edge device of the subsystem, to eventually deploy the software artefacts according to the deployment model. Within a fleet, the subsystems have different contexts, such as the device capacity, the connectivity, the user preferences, etc., and developers need multiple variants of their software to fit different contexts. DivEnact accepts multiple deployment models representing different software variants and configurations, coming for a series of releases, and automatically assign them to the proper subsystems. For the sake of availability, we recommend running the main service of DivEnact in the Cloud, with a light-weight DivEnact broker running on edge devices of each subsystem.

Next, we present GeneSIS and DivEnact, detailing their main innovations as well as how they contribute ensuring the trustworthiness of SIS.

4.3.1 GeneSIS

GeneSIS enables the continuous orchestration and deployment of Smart IoT Systems throughout the IoT-Edge-Cloud continuum. Given a description of a deployment topology, GeneSIS deploys and configures the needed software components, by connecting to the hardware (or software) nodes. This topology, the so-called deployment model, only prescribes what components must be deployed, how a single component can be deployed, and how they connect to each other. GeneSIS automatically derives how to deploy them. Therefore, GeneSIS is composed of two key components: (i) a domain-specific modelling language for specifying deployment models, and (ii) an execution engine to enact the provisioning, deployment and adaptation of a SIS. We refer the reader to [20] for more details about the GeneSIS modelling language.

The target user groups of GeneSIS are mainly DevOps engineers, software developers, and software architects. The GeneSIS modelling language has been conceived so the deployment model can act as a touch point between development and operation activities. DevOps teams can use it to deploy either in development, staging or production environments. It is also worth noting a deployment model written using 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.

The main task of the GeneSIS deployment engine is to reconcile two views of the system: the deployment model given by the user, and the current state of the running infrastructure, assuming that software components may already be running on the infrastructure, for example, as a result of a system upgrade. To reconcile these two views, the GeneSIS deployment engine adheres to the "models@runtime" architectural pattern [6]. It compares these two views and deduces what changes the adaptation engine must carry out on the running infrastructure to align it with the prescription, i.e., the deployment model given by the user. After the deployment, the engine synchronizes the current GeneSIS model with the actual deployment result. Such synchronization will ensure that all the tools in future DevOps cycles will leverage an up-to-date deployment model.

The GeneSIS deployment engine is non-invasive, meaning it does not require any GeneSIS bootstrap or agent running on a target device to deploy software on it. However, when decided by the DevOps engineer, the GeneSIS deployment engine can deploy on a target device a monitoring agent. This agent is an instance of netdata[10] and provides information about the performance and health status of a device, including data about software components it hosts.

Finally, the deployment engine can delegate parts of its activities to deployment agents running in the field. It is not always possible for the GeneSIS deployment engine to directly deploy software on all hosts. For instances, tiny devices do not always have direct access to the Internet or even the necessary facilities for remote access (in such case, the access to the Internet is typically granted via a gateway) or for specific reasons (*e.g.*, security) the deployment of software components can

10. https://github.com/netdata/netdata

only be performed via a local connection (*e.g.*, a physical connection via a serial port). In such case, the actual deployment of the software on the device has to be delegated to the gateway locally connected to the device. The GeneSIS deployment agent aims at addressing this issue. It is generated dynamically by GeneSIS based on the artefact to be deployed and its target host, and is implemented as a Node-RED application. We refer the reader to [20, 21] for more details.

GeneSIS comes with a set of predefined component types that can be seamlessly instantiated in deployment models. In addition, GeneSIS embeds a plugin mechanism that enables the dynamic loading of new component types. A components type repository is scanned by the GeneSIS execution engine before each deployment ensuring all available types are loaded before a deployment model is analyzed and deployed.

4.3.2 DivEnact

While GeneSIS focus on the deployment of a single system, DivEnact, the diversity-oriented fleet deployment enabler, is an implementation of our concept of *fleet deployment*. Fleet deployment is an automatic software deployment support for IoT/Edge applications, which allows developers to deploy software artefacts onto a fleet of devices as an abstract whole, without concerning about the concrete devices and their contexts in the fleet. The automatic fleet deployment tool, such as DivEnact, will maintain the devices and their contexts in the fleet, the software variants, and assign the variants to the appropriate devices depending on their contexts.

DivEnact utilizes Azure IoT Edge to maintain a list of edge devices, together with their contexts and run-time status. Developers provide DivEnact with a set of deployment models (typically GeneSIS models), each of which specifies a particular software artefact, together with the specification about how to configure and deploy it on an Edge device. In order to facilitate the definition of similar deployment models, we also introduce the concept of deployment *templates* and *variants*. A template defines the common parts among a number of deployment models, and a variant further instantiates the template as a deployment model. A common use case for this is to define a deployment model for a particular software, and then use variants to represent the different versions of this software. After receiving all the deployment models, DivEnact automatically assigns them to the list of edge devices, and enacts the deployment model on each edge devices to finalize the local deployment.

Figure 4.2 illustrates the technical architecture of the DivEnact tool. The DivEnact tool is designed and implemented following the established Model-View-Controller (MVC) design pattern for client-server application systems.

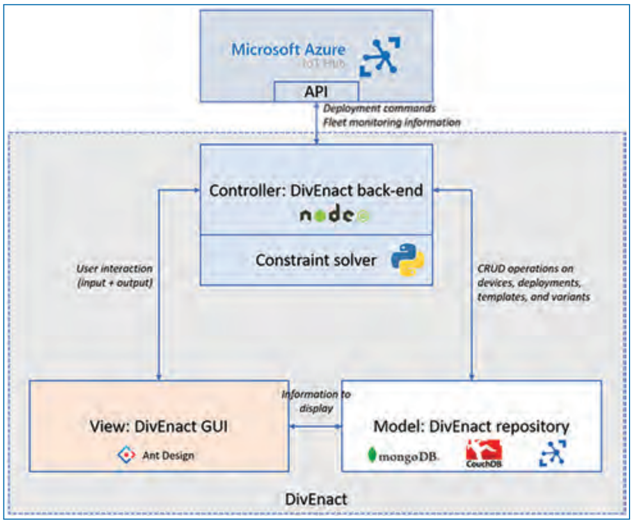

Microsoft Azure
IoT Hub
API
Deployment commands
Fleet monitoring information
Controller: DivEnact back-end
node
Constraint solver
User interaction (input + output)
CRUD operations on devices, deployments, templates, and variants
Information to display
View: DivEnact GUI
Ant Design
Model: DivEnact repository
mongoDB
CouchDB
DivEnact

Figure 4.2. The architecture of DivEnact.

The DivEnact knowledge base stores various deployment- and fleet-related artefacts and is modified by the DivEnact back-end upon the user input received through the graphical user interface. The modification actions (CRUD – create, read, update, delete) are implemented on top of standard APIs and libraries. The model itself is spread across the following three repositories. MongoDB database is installed locally, along with a DivEnact instance, and serves to store information about templates and variants unique to each application system. There is a centralised repository in CouchDB for storing various ENACT artefacts, including deployment models used by DivEnact. In particular, the CouchDB database stores previously designed GeneSIS deployment models that are to be enacted on low-level IoT devices as part of the "last mile deployment". Azure IoT Hub Cloud portal keeps track of registered devices in the fleet and existing deployments. The information obtained from the hub reflects the current state of all the devices through continuously updated digital twins, as well as deployments applied to these devices (i.e., software modules currently deployed and running on each device).

The DivEnact graphical user interface remains the main point of interaction with the user. The main functionality is structured across several functions, i.e., the editing and maintenance of templates, variants, deployment models, devices and the assignment.

The back end of the DivEnact tool is implemented in Node.js. It receives RESTful requests originating from the user's graphical interface and manipulates the data model accordingly. It also interacts with the Azure IoT Hub API to update some information about the devices in the fleet and trigger deployments. The back end also implements the actual diversification functionality (described in the next subsections) by receiving the input model from the user and passing it to the underlying Python script. Upon execution, the calculated solution is passed back to the user for the final approval.

The main function of the back end is the automatic assignment of deployment models into the list of devices, considering the constraints, the deployment preferences, the resource optimization, etc. We have implemented two experimental assignment approaches, using constraint solving and resource assignment theories as the back end mechanisms. The details of these two approaches can be found in our recent publications [15, 44].

4.4 Trustworthy Deployment

As explained before, ensuring the trustworthiness of the deployment and of the SIS is critical and challenging. It is a concern that crosscuts both the device or at fleet layers. In the following we detail how GeneSIS and DivEnact help addressing this challenge. Our effort is concentrated on three complementary directions, i.e., how to increase the availability of deployed system; how to automatically deploy the required security mechanisms together with the application; and how to maintain the software diversity across the whole fleet.

4.4.1 Deploying Availability Mechanisms

Availability refers to "the ability of the system to mask or repair faults such as the cumulative service outage period does not exceed a required value over a specified time interval" [4, p. 174]. Availability is a primary concern for business stakeholders because service interruptions often translate into money loss. The failure of an electricity meter for instance may affect the capacity of the electricity company to properly bill its customers.

Availability, as any extra-functional requirements, does not affect the system function, but rather affects its architecture. Building high-availability systems requires additional components to detect, repair, or even prevent faults, such as monitors, watchdogs, replicas, or voting mechanisms to name a few. Availability tactics are now well documented, so we refer the reader to [4, Chap. 5] for an introduction.

Many things can go wrong in Smart IoT Systems, including incorrect algorithms, network failure and delays, hardware failure, etc. In the following, we focus on scheduled outages, which are interruptions of service needed because of software upgrade, and internal faults, which are faults that occur because of defects in the source code of the components. In other words, the availability support we present hereafter contributes (i) improving trustworthiness of a SIS by maximising its availability (by minimizing downtime during upgrades); and (ii) improving deployment trustworthiness by modifying a system and its deployment only if the deployment process is successful (i.e., old version of a software is removed only if the new version is up and running).

We extended GeneSIS with the ability to deploy mechanisms that cope with these two kinds of fault. To mask internal faults, GeneSIS deploys multiple instances of the same service/component (so called replicas) behind a proxy. When one replica fails, the proxy can query another replica. To mask scheduled outages, GeneSIS provides zero-downtime upgrades. We leverage the same architecture and deploys the new version (behind the proxy) before to decommission the older one. That way, there is always a replica available and upgrades do not affect availability.

Modern execution platforms such as Docker or AWS already implement various availability mechanisms, and, they have strategies for both scheduled outages and fault-tolerance. Docker Swarm for instance performs zero-downtime upgrades by deploying new services instances before to decommission the older ones. The challenge is that, from a deployment perspective, the availability tactics are tightly coupled to the underlying execution platform. Changing platform requires changing the deployment configuration.

To decouple availability from deployment platform, GeneSIS captures these deployment tactics independently of the underlying platform. If the platform already provides mechanisms (such as Docker Swarm), GeneSIS uses those, otherwise it deploys built-in components to implement the selected tactics. In the following, we illustrate three scenarios that show how GeneSIS copes with scheduled outages and internal faults.

1. Initial Deployment: GeneSIS deploys the system following the availability tactics selected by the user.
2. Internal Fault: A fault occurs in the system and we explain how the mechanisms that GeneSIS has deployed deal with that fault, so that it is not visible to the end-user.
3. Zero-downtime Upgrades: The user requests the deployment of a new version of the system and we illustrate how GeneSIS leverage the underlying mechanisms to minimize service disruption.

4.4.1.1 Using built-in components on top of docker

By default, GeneSIS does not make any assumption of the capability of the platform where it should install a component. It could be a very fully featured platform such as Docker Swarm (see Section 4.4.1.2) or simply an operating system offering remote access (through SSH, Telnet, etc.). We detail here the later case, that is when the host is a bare OS. Recall that GeneSIS uses two strategies to improve availability: Replication to deal with internal faults, and zero-downtime deployment to deal with scheduled outages. To implement these two strategies, we need three capabilities that are provided by additional components:

- *Routing*, that is, the capability of redirecting incoming traffic to a selected replica. Network proxies provide this and in GeneSIS, we selected Nginx.
- *Error detection*, that is, the capability to proactively detect replicas that have failed (for whatever reasons). We used a watchdog, that is a component that periodically connects to the replicas and runs a so-called "health check". The health check is an application specific behaviour that confirms that the replicas is up and running. It could be requesting a predefined resource using HTTP, checking the status or OS-level services, or any other "quick-check". In GeneSIS, we have implemented simple watchdogs using Shell scripts and CRON tasks.
- *Spatial isolation*, that is, the capability to deploy multiple instances of the same application with guarantees that they can access external resources (network port, files on disks, etc.) without stepping on each other. GeneSIS uses containers (*i.e.* Docker in the current implementation) to ensure spatial isolation of replicas, but other container technologies such as LXC apply.

Scenario 1: Initial Deployment

The first step is for GeneSIS to ensure that the underlying host offers "spatial isolation" guarantees. To do so, GeneSIS first installs Docker as container offer such guarantees. Figure 4.3 below illustrates how GeneSIS interacts with the host to install docker and to create a "replicable image" of the software stack.

Given a component to deploy, GeneSIS first connects to the host through SSH and installs Docker (Step 1). Then GeneSIS configures Docker in remote mode so that other components (including itself) can access it through the network. Then, GeneSIS creates a new temporary container (by default, using the image "debian:10-slim") and installs the underlying software stack. To do this, GeneSIS traverses the underlying software stack and installs all underlying components by triggering the associated SSH commands into the container (Step 6, 7 and 8). Once the stack is installed and configured, GeneSIS converts it to a separate Docker image, that it later uses to install multiple replicas (Step 9). Finally, GeneSIS destroy the temporary container. At this stage, GeneSIS has enforced spatial isolation, and can then proceeds with replication and zero-downtime upgrades.

Once Docker is operational and the component to install is available as a Docker image, GeneSIS proceeds with the two remaining capabilities, namely

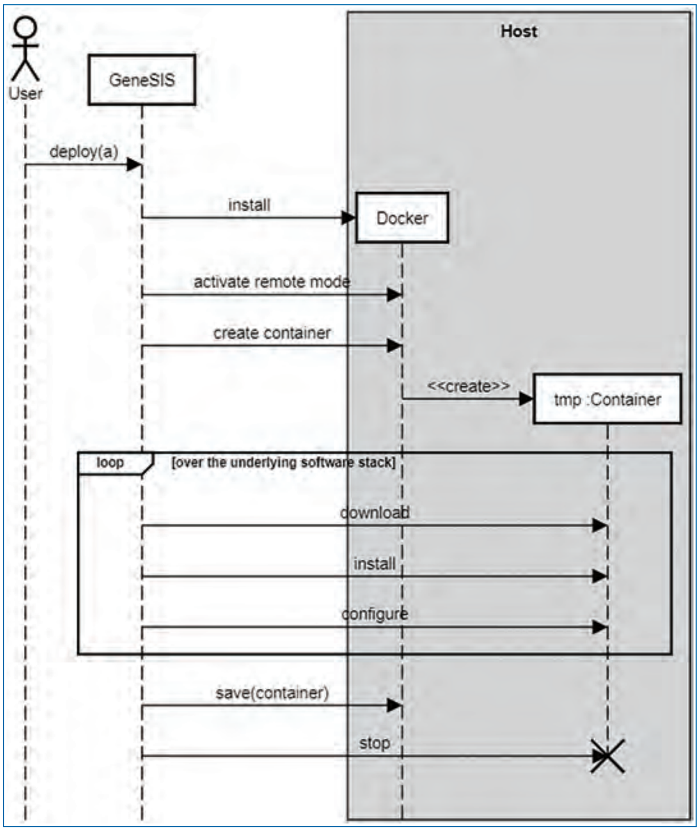


Figure 4.3. Automatically converting SSH resources into a Docker image. GeneSIS connects to the host, and execute all SSH commands into a new container, which it then saves as a new “ready to use” image.

detecting errors and routing as shown on Figure 4.4. First GeneSIS installs the proxy component through Docker (Step 1 and 2). Then, it installs the watchdog and configures it with the endpoints of the Docker host, the proxy and with the number of replicas to maintain (Step 2 and 3). For each missing replica, the watchdog requests Docker to provision a new instance of the image built in Scenario 1 and then the watchdog start checking the health of each replica periodically (Step 6 and 7). As soon as a replica is detected as healthy, the watchdog registers it to the proxy (Step 8), which uses it process user requests (Step 9 and 10).

Scenario 2: Fault tolerance

We now turn to the second scenario where one replica fails and we explain on Figure 4.5 how the watchdog detects and reacts to such a failure. The main mechanism to detect failure is the health check. Since a health check is an application-dependent behaviour, the user must provide it as a script to be executed periodically

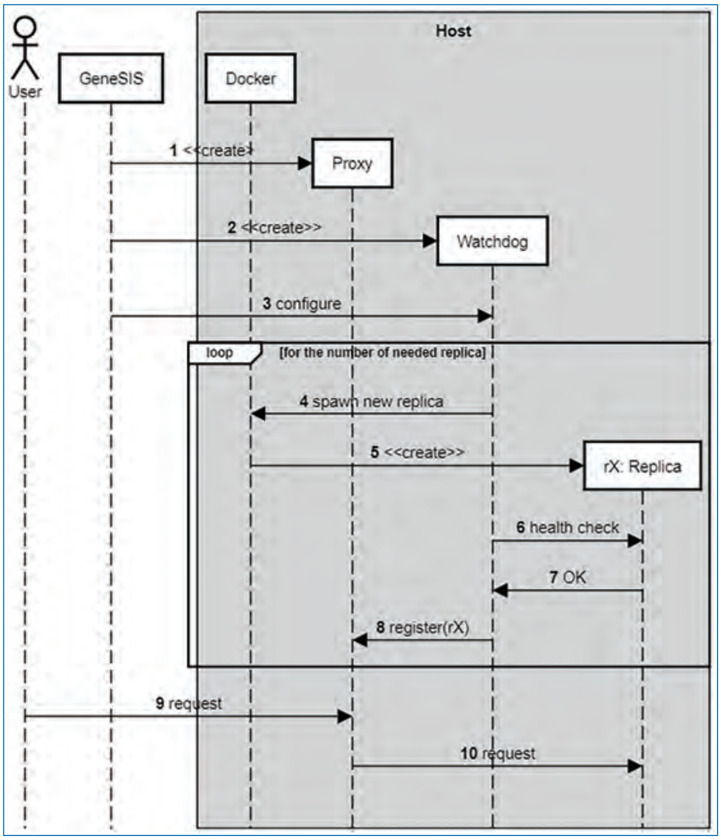


Figure 4.4. Configuring watchdogs and proxies to improve availability.

by the watchdog. GeneSIS defines the interface of the health check script as follows. The health check must accept the endpoint of the replica to query as its sole input parameter and must output the replica status in return through its exit code: Zero if the replica is healthy and any other value otherwise. This gives the user the capability to integrate any application-specific health check logic. The listing below shows one such health check script based on the HTTP status code, returned by a service.

```
#!/bin/bash
ENDPOINT="${1}"
response=$(curl --write-out '%{http_code}' --silent --output /dev/null"
    ${ENDPOINT}")
 if [ "${response}" != 200 ]
then
    exit 1
 fi
```

Listing 4.1. A Sample health-check script.

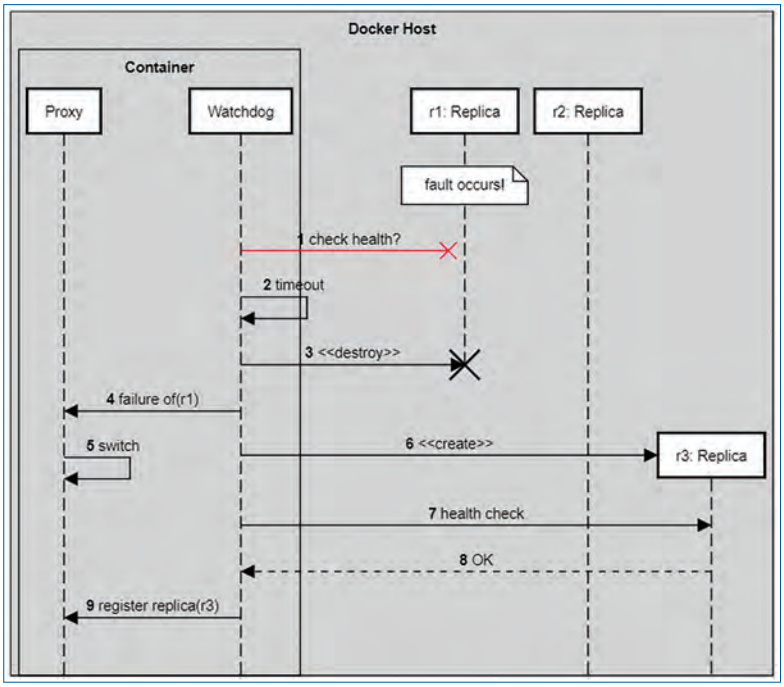


Figure 4.5. Masking internal faults to improve availability.

Note that this architecture can only detect replicas' failure as fast as the watchdog waits between two health checks. Besides, for the failure to be invisible to the user, there must at least two replicas, for the proxy to switch between them as soon as a delegation fails. Finally, transient phenomena such as network delays may be mistaken for replica failures and lead to unnecessary starts and stops of the container.

Scenario 3: Zero-downtime Upgrades

Finally, GeneSIS leverages these proxy and watchdog to guarantee zero-downtime upgrades, as shown on Figure 4.6.

When the user requests an upgrade, that is the deployment of a new version, GeneSIS first builds a new docker image of the software stack, including this new version. We described this process in Figure 15. Once this new image is ready, GeneSIS request the watchdog to perform the upgrade (Step 2). The watchdog thus provisions new instance of the new version (Steps 3 and 4) and, once these new replicas are operational, the watchdog registers them to the proxy (Steps 4, 5, 6 and 7). At that stage, incoming requests from the user are still delegated to the older version (Steps 8 and 9). Only once all replicas of the new version is operational, then the watchdog starts to decommission the older versions (Steps 10 and 11).

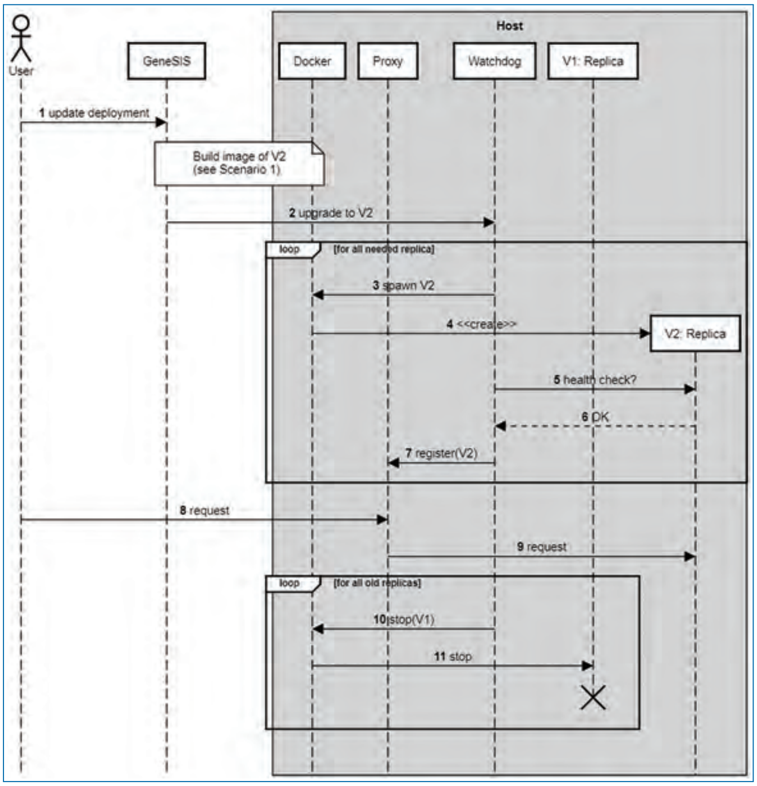


Figure 4.6. Using proxy and watchdog to guarantee zero-downtime upgrades.

4.4.1.2 Using docker swarm

In many cases, developers do not choose the platform on which their software runs: It may result from organizations' policies, customer requirements, etc. Platform such as Kubernetes, Docker Swarm or Rancher for instance all implement availability tactics, including replication and zero-downtime releases. Should the user use such platform, GENESIS can exploit these native features to ensure fault-tolerance and zero-downtime upgrades. Docker swarm already implements routing among multiple replicas and fault detection. GENESIS therefore delegates these features to Docker Swarm. We briefly example how GENESIS handles our three scenarios using Docker Swarm.

Scenario 1: Initial deployment

This step is the simplest as we assume here that the host already runs Docker Swarm and that it therefore already guarantees spatial isolation. Here, GENESIS simply requests Docker Swarm to deploy the given number of replicas of a given Docker image.

Scenario 2: Fault tolerance

This is also fully transparent from the GeneSIS standpoint. When GeneSIS delegates the deployment to Docker Swarm it specifies the number of replicas and the health check script to be used to detect faults. It is docker swarm that periodically checks the replicas status, provisions new ones if some have failed and route incoming requests accordingly.

Scenario 3: Zero-downtime redeployment

Further Docker Swarm also offers various strategies to upgrade a service (*i.e.*, the set of replicas in Docker Swarm parlance). Among many options, Docker Swarm lets the user specify the "update order". If this order is "stop-first", then Docker Swarm first stops all the replicas of the older version, and only then starts provisioning replica for the new version. By contrast, if the update-order is "start-first", then Docker Swarm provisions all new replicas before to decommissions, as GeneSIS would do without Docker Swarm (see Figure 18). This "start-first" option let Docker Swarm minimize service interruptions.

4.4.1.3 Limitations

The support for availability tactics in GeneSIS is limited to Docker platform, although the general principle applies regardless of the underlying technology and other can extend GeneSIS and support other technologies. In addition, there are other types of faults that the current tactics cannot deal with. Hardware failure for instance would take down the whole host and therefore all the replicas at once. To tackle hardware failure, replication would have encompassed hardware, but this goes beyond GeneSIS whose mission is to provide platform agnostic deployment. Nevertheless, using the ENACT framework, the Root Cause Analysis enabler can be used to monitor and identify such failures and DevOps engineers can use GeneSIS to migrate the software components on a new host, benefiting from its platform independence. Programming faults are also not dealt with. Because GeneSIS is oblivious to the inner working of the components it deploys, all replicas are similar and fail in a similar manner. For instance, if a defect in the code lead to a fault of one replica (say because of invalid user input), then all replicas will exhibit this fault. Only diversification techniques [5] could help having replicas whose behaviours differ from one another, and that exhibit different failure profiles.

Improving availability from a pure deployment perspective, as GeneSIS provides, is bound to stateless components that can be easily replicated. Replicating a component that persists state requires some modification of its code. Either we separate its state from its application logic (using a local database, for instance) and we ensure that all replica can access this single data source. Alternatively, each replicas also have its own local copy but we must now define a strategy to ensure the correct and timely synchronization of the multiple copies of the state, and possible conflicts. Modern database engines offer such mechanisms and would need to be integrated with GeneSIS in an ad-hoc manner. Edge platforms however often only pass data further on to Cloud services, and are thus likely to be stateless, or can simply leverage a local database as a cache, a strategy that the GeneSIS availability mechanisms handles.

Finally, on an Edge platform there are resources that cannot be replicated and that would require further investigation. A serial link for instance cannot be shared between replicas and, in this case, dedicated, application-specific logic must be in place to ensure consistent behaviour between all the replicas.

4.4.2 GeneSIS for Continuous Deployment Supporting DevSecOps

GeneSIS empowers a DevOps team to cope with security and privacy concerns of SIS as it natively offers support for including, as part of the deployment models, concepts to express security and privacy requirements and for the automatic deployment of the associated security mechanisms [20]. More importantly, GeneSIS enables the continuous enhancement of security controls in a DevSecOps cycle to keep security mechanisms up-to-date and well-aligned with the evolution of SIS, as well as addressing IoT security risks that are always evolving.

In this sub-section, we present the latest development of GeneSIS for better supporting the continuous deployment and enhancement of security controls that can refine or override the associated (default) security and privacy mechanisms of the IoT platforms such as SMOOL [9] or FIWARE [11]. Such security mechanisms are further elaborated in Chapter 7. More specifically, GeneSIS provides a generic way for a DevOps team to extend such existing security mechanisms with other (third-party) security mechanisms to provide enhanced security controls in a DevSecOps fashion.

4.4.2.1 GeneSIS for the specification and deployment of security components

To better support DevSecOps, GeneSIS promotes specifying security mechanisms as explicit elements in the deployment model, instead of hidden (and thus tightly coupled) in the source code, so that developers can see and change the security mechanisms in the deployment model level. This includes specifying security requirements and capabilities, and supporting the deployment of security mechanisms as components reusable in different scenarios. Compared to the previous GeneSIS version reported in [21], we have built a new library of off-the-shelf security components that can be selected for instantiating in the deployment model. More importantly, we provide DevOps teams with mechanisms to configure security components and inject fine-grained security policies into deployment components (without modifying their business logic), enabling their seamless integration with third party security mechanisms (services, libraries, etc.). These supports can ease the development, integration, and deployment of SIS with continuously enhanced security mechanisms (see Section 4.4.2.2).

GeneSIS supports the deployment of security components as any other software components in the way that their deployment and configuration can be defined via exposed APIs and configuration files. A security component to be deployed together with an IoT application can be declared in GeneSIS with "security capabilities" in a provided port. A required port of a software component that requires a matching security capability can be bound with the provided port of the security component that provides such security capability. Before enacting a deployment, the GeneSIS deployment engine validates the correctness of the provided deployment model. In particular, it ensures that the required "security capabilities" match the provided ones.

GeneSIS allows specifying the deployment of security mechanisms and policies built on top of IoT platforms. We present here its application to the SMOOL IoT platform, which is used in our ENACT project. Similar approach can be applied to other IoT platforms. At the development phase (as well as at the deployment phase presented below), GeneSIS provides the support to relieve developers from manually specifying and maintaining security monitoring and control mechanisms in the code of a SMOOL client. Instead, a developer can define its own SMOOL client, focusing on its business logic. We integrated the SMOOL client wizard with ThingML[11]. As a result, a single Eclipse IDE can be used to generate the code of a SMOOL client, which can then be directly used as part of a ThingML program. The proper Maven manifests are automatically created facilitating the building and release of the desired application. This means that the DevOps team can quickly develop the business logic of the SIS based on the SMOOL platform, including necessary security mechanisms. DevOps teams can define SMOOL clients that leverage built-in security properties to check and enforce security concepts on messages requiring security controls.

The SMOOL's default security enforcement can be done with the SMOOL clients built-in security metadata checker to verify messages exchanged

11. https://github.com/TelluIoT/ThingML

among them. In cases where a deeper control is needed, a specialised security metadata checker can be included in SMOOL clients, with additional privileges to watch and process the security metadata in messages exchanged, in the same way it is done with business logic concepts such as sensed temperature or gas values. This provides a fine-grained control on critical messages that may have a significant security impact in the IoT system such as orders to actuators. More precisely, a client code can conduct security checks based on policies to be fulfilled by ontology concepts by using any of these options: (i) the default security metadata checker (for minimal configuration), (ii) a custom security metadata checker implemented in the development phase (for full control of security), and (iii) a custom security metadata checker for integration with external security services. Whatever security options, GeneSIS provides support for easily configuring the security mechanisms and how they should be integrated and deployed with the SIS. Thanks to ThingML, GeneSIS provides advanced support for the three options.

To support the first option, GeneSIS enables the DevOps team to specify explicitly the default security policy that must be enforced by the Security checker. To support the second option, where the DevOps team can implement its own ad-hoc security checker, GeneSIS provides the means to automatically inject this security checker into the code of the component to be deployed and to rebuild the component automatically. More precisely, when deploying this security component, the GeneSIS deployment engine injects the security policy into the ThingML code of the SMOOL client. This code injection is done before GeneSIS triggers the compilation of this code to generate the actual implementation of the SMOOL client with the corresponding security policy.

To support the third option, GeneSIS not only injects the security checker code that integrates with a third party security solution (*e.g.*, Casbin[12] or the Context-aware Access Control mechanism [23], or a "gatekeeper" in [33]) but it can also deploy the latter. At the deployment phase, a SMOOL client can be deployed by GeneSIS as any other software components. Once the SMOOL client has been developed, the developer can specify how to deploy it together with the security and monitoring mechanisms that should apply to its SMOOL client. GeneSIS will then inject within the SMOOL client the necessary code to perform the security checks before actually deploying it. To do so, we created a generic security component that represents a SMOOL client as a deployable artefact. This client can follow any of the security check options discussed above and is implemented with ThingML code, which integrates (i) the necessary SMOOL libraries, (ii) the SMOOL client business logic, (iii) and the security logic. The main rationale behind this choice is

12. https://casbin.org/

the following. ThingML offers an extra abstraction layer that provides the ability to wrap the code and dependencies that compose a SMOOL client and to inject into it the necessary security code. In addition, it provides GeneSIS with a standard and platform-independent procedure to generate, compile, configure, and deploy the implementation of the security mechanisms. A similar approach could be applied to other IoT platforms. In this way, GeneSIS allows DevOps teams to reconfigure and update security mechanisms by design, in line with the evolution of IoT applications and the development of security and privacy risks. In the next section, we present more details on the DevSecOps support.

4.4.2.2 The DevSecOps support for the continuous enhancement of security mechanisms

SIS typically expose a broad attack surface and their security must not be an afterthought [22]. The ability to continuously evolve and adapt these systems to their dynamic environment is decisive to ensure and increase their trustworthiness, quality, and user experience. This includes security mechanisms, which must evolve along with the SIS, continuously fixing security defects and dealing with new security threats [32, 35]. Following the DevSecOps principles [30], there is an urgent need for supporting the continuous deployment of SIS, including security mechanisms, over IoT, Edge, and Cloud infrastructures [1]. The DevOps movement promotes an iterative and incremental approach enabling the continuous evolution of software systems. As an evolution of the DevOps movement, DevSecOps promotes security as an aspect that must be carefully considered in all the development and operation phases for the continuous evolution of systems to be secure.

In this section, we present how GeneSIS can enable the continuous enhancement of security controls in a DevSecOps cycle: from development to operation. GeneSIS also supports the adaptation of the system having enhanced security mechanisms or updated security policies with minimal impact on the already delivered and under operation. Our approach [18, 20, 21] for the continuous deployment of SIS with enhanced security mechanisms can serve the DevOps team in both adaptation and evolution of the SIS. First, GeneSIS supports for evolving SIS with updated security mechanisms according to a new development cycle. Second, GeneSIS supports for adapting security enforcement to improve how the IoT system operates securely. This DevSecOps support leverages the GeneSIS ’ necessary mechanisms, interfaces, and abstractions to dynamically adapt the deployment and configuration of a SIS as presented earlier. We elaborate more on the two kinds of DevSecOps support in the following paragraphs.

First, GeneSIS supports for evolving SIS with updated security mechanisms according to a new development cycle. In this line of adaptation, the SIS in operation is evolving with new business logic components or even new physical devices

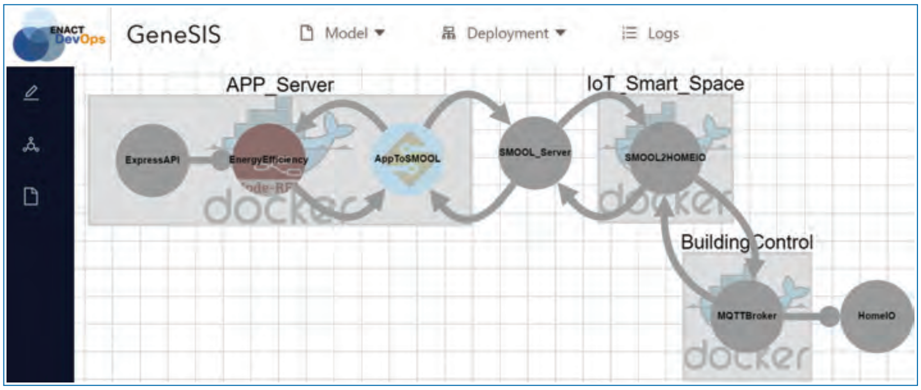


Figure 4.7. An initial version of a smart home system (deployment view).

being added resulting in the need for enhancing security mechanisms accordingly. We demonstrate this support using a smart home simulation called HomeIO.[13] More details on the deployment demo using the HomeIO simulation can be found in this video.[14] In the smart home system, there are IoT applications (*e.g.*, *UserComfortApp*) that get access to sensors' data (*e.g.*, temperature) from the smart home to make decisions and send commands to control the actuators, *e.g.*, window blinds. The applications interact with the smart home devices and services via the SMOOL platform (in the middle of Fig. 4.7). GeneSIS can easily support for the deployment of components that are either built on top of the existing IoT platforms like SMOOL or are independent of any IoT platform because of its generic approach for specifying deployment components. However, to make GeneSIS even more useful in practice, we have developed GeneSIS to ease the integration of IoT platform-specific components (*e.g.*, SMOOL clients) and IoT platform-independent components (*e.g.*, third-party security mechanisms like Casbin presented below) from development to operation.

In the initial version of the smart home system, there is the *EnergyEfficiency* application, which gets access to sensors' data to make decisions for energy efficiency and send commands to control the actuators, *e.g.*, window blinds. In particular, it maximizes the exploitation of daylights and regulates the in-door temperature whilst minimizing the energy consumption. If the room is bright because of daylight, it will switch off the LED-lights, and vice versa. On the other hand, if the room temperature is high, the application may need to close the window blinds to

13. https://realgames.co/home-io/

14. https://youtu.be/yQ9XYWu-EZM

prevent sunlight heating the room. The *EnergyEfficiency* application interacts with the smart home devices via the SMOOL platform. There are two notable security mechanisms associated in this first version of the smart home. The first one is a secure API gateway (Express Gateway[15]) that allows secure remote API access to the *EnergyEfficiency* application. The second one is a *SecurityEnforcer* by default of the SMOOL middleware that enforcing the security check for the data passing through, *e.g.*, only allowing genuine actuation commands to be sent to the actuators of the smart home. The latest version of GeneSIS has provided a built-in support to ease the specification of the Express API Gateway in the deployment model. Adding a new instance of Express API Gateway is easy. The remaining work for the DevOps team is to specify the configuration files of the API gateway, which define how the API of the *EnergyEfficiency* application can be securely accessed.

In IoT platforms like SMOOL, there are often default security enforcements. For example, the actuation orders must be checked before they are actually sent to the actuators. This check (embedded in the *SMOOL2HOMEIO* component, Fig. 4.7) makes sure only genuine actuation commands can be sent to the actuators. In other words, the SMOOL platform allows to check for actuation commands with valid security tokens. All the IoT apps must send actuation commands with valid security tokens.

However, during the evolution of the smart building system, new applications can be added, and new physical devices can also be added. In the subsequent development cycle, another application called *UserComfortApp* has been added to the smart home system. Moreover, the smart home system can also have new IoT devices such as *AirQualitySensor* or *SmartDisplay* as shown in Fig. 4.8.

New security requirements come up because the smart building system must control which apps can access which actuators. This means that more fine-grained security control must be introduced, which may not be available in the IoT platform. GeneSIS should support for seamlessly integrating new (third-party) security mechanisms into the IoT platforms. In this new development cycle, not only that the secure API gateway must be updated with a new configuration file, but also the DevOps team needs to introduce a new security mechanism that can enhance the fine-grained control of how different applications can access to the sensors and actuators of the smart home system. GeneSIS has a generic support for seamlessly integrating and deploying any advanced security mechanism together with the IoT platform in use, *e.g.*, the SMOOL platform. More specifically, in this example, the DevOps team develop an access control mechanism based on an open source

15. https://www.express-gateway.io/

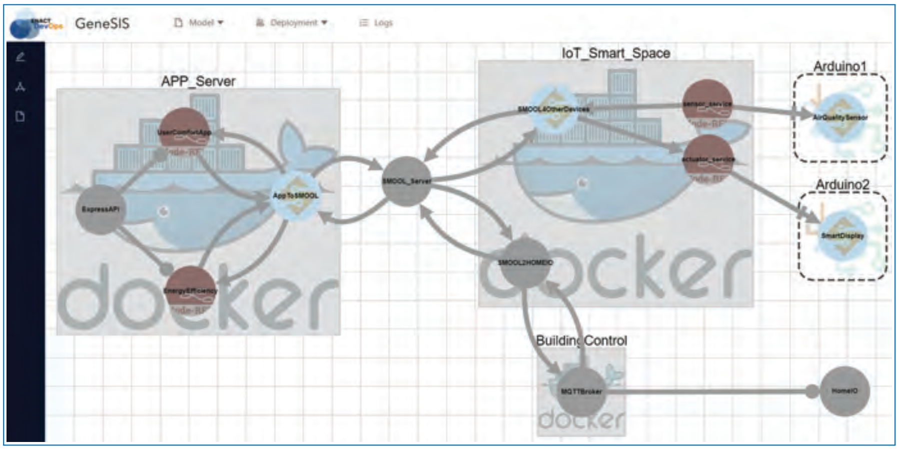


Figure 4.8. New applications and new IoT devices can be added in a development cycle.

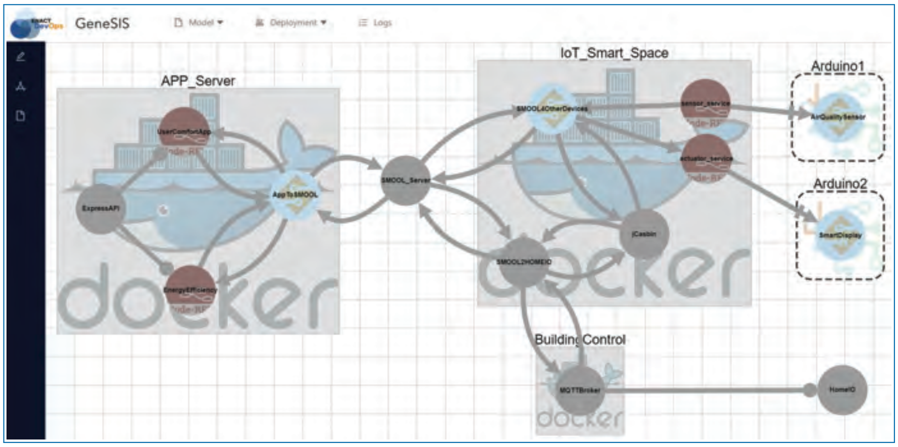
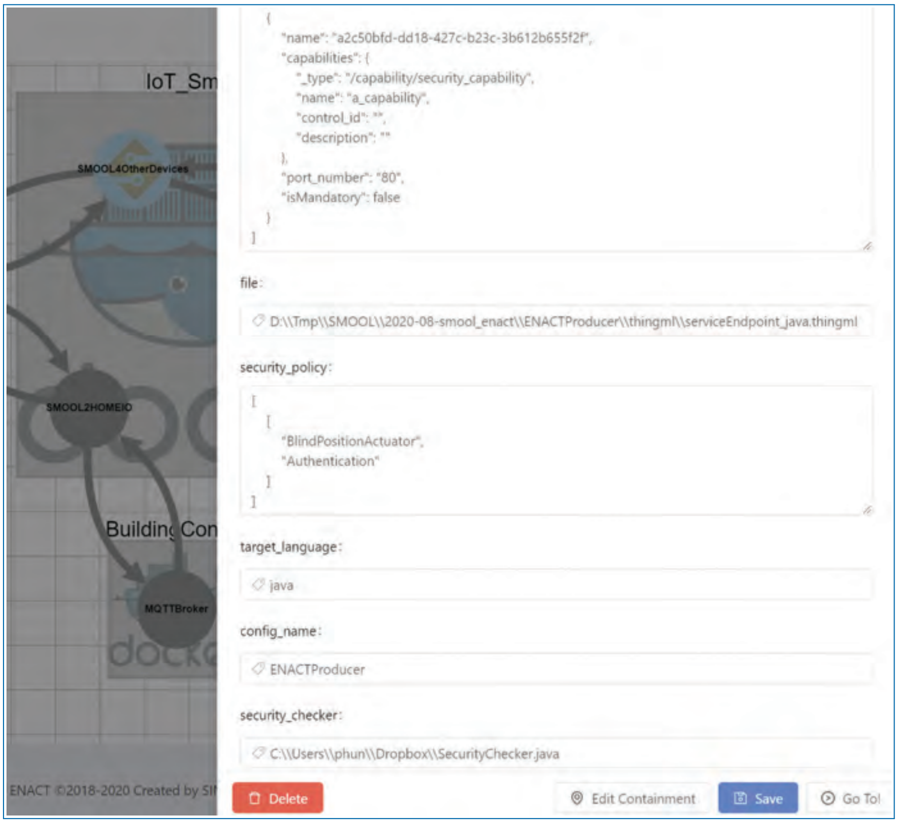


Figure 4.9. An enhanced security control has been added.

framework called jCasbin,[16] and then specify the integration point with the IoT platform in use (with GENESIS support, see Fig. 4.9). During the deployment process, GENESIS compiles the integration code before orchestrating the deployment of the integrated components.

To enable such DevSecOps adaptation support, GENESIS not only provides the modelling language embedded in a web UI for specifying the components of such IoT platforms, but also the reconfiguration and rebuild of these components (for integrating new security mechanisms with the IoT platform) before deployment (for adaptation or for a new development cycle). For example, in the SMOOL

16. https://casbin.org/



```
    {
      "name": "a2c50bfd-dd18-427c-b23c-3b612b655f2f",
      "capabilities": {
        "_type": "/capability/security_capability",
        "name": "a_capability",
        "control_id": "",
        "description": ""
      },
      "port_number": "80",
      "isMandatory": false
    }
]
```

file:

D:\\Tmp\\SMOOL\\2020-08-smool_enact\\ENACTProducer\\thingml\\serviceEndpoint_java.thingml

security_policy:

```
[
  [
    "BlindPositionActuator",
    "Authentication"
  ]
]
```

target_language:

java

config_name:

ENACTProducer

security_checker:

C:\\Users\\phun\\Dropbox\\SecurityChecker.java

Figure 4.10. An enhanced security control has been added.

platform, each SMOOL producer or consumer is associated with a security checker for checking the security key of sensor data or actuation commands. GENESIS allows updating the configuration of the security checker (*e.g.*, by injecting new configuration to overwrite the default one), and automatically rebuilding the SMOOL producer or consumer including the reconfigured security checker. By doing so, GENESIS enables the DevOps team to make reconfiguration or redevelopment and redeployment easily for the evolution of SMOOL producers or consumers including security checkers. Figure 4.10 shows an example of the GENESIS's UI for extending the *SecurityChecker* (in *SMOOL2HOMEIO*) to become a security enforcement point of the external access control service. Thanks to ThingML support within GENESIS, the extended *SecurityChecker* is compiled in the *SMOOL2HOMEIO* component for a new version of *SMOOL2HOMEIO* to be deployed that works as a security enforcement point of the external access control service.

This approach is what we call the DevSecOps adaptation support for the co-evolution of business logic components and the security mechanisms. This means that when new business logic components require security mechanisms to evolve, GeneSIS can support for the adaptation, even including the integration of the IoT platform with other (third-party) security mechanisms.

After the successful deployment, GeneSIS allows dynamic adaptations that can be triggered at any point, manually or automatically for adapting security enforcement to improve how the IoT system operates securely. In this line of adaptation, new security policies or configurations can be updated dynamically for the security mechanisms that are in operation. For example, the role-based access control policy can be easily updated according to new requirements. The trigger of such adaptation can be manually, but also can be automatically from a risk assessment process or after a reasoning process of actuation conflict management.

In summary, with the support from GeneSIS, the DevOps team can develop a new version of the smart home system together with enhanced security mechanisms according to its evolution. The deployment of this new development cycle can be triggered manually from GeneSIS's GUI. After the successful deployment, GeneSIS also allows dynamic adaptations that can be triggered at any point, manually or automatically for adapting security enforcement to improve how the IoT system operates securely. In both ways presented so far, GeneSIS allows DevSecOps teams to reconfigure and update security mechanisms by design, in line with the evolution of IoT applications and the development of security and privacy risks.

It is important to note that in this chapter we have not addressed the security of the build and deployment pipeline itself. The security of this pipeline is critical to protect the integrity of the code and the systems being deployed. For the production environment, GeneSIS must adhere to the secure deployment practice.[17] One of the main principles in secure deployment is to support automatic testing as part of the deployments to gain confidence in the security of the code (see Section 8.2 for Test and Simulation).

4.4.3 Software Diversity Within IoT Fleet

Software diversity in an IoT fleet, i.e., deploying variants of software on different devices, creates a moving target for malicious attacks, and therefore improves the overall security of the system. The DivEnact tool assigns the available variants to the fleet of devices and maintains the balance between the variants. The remaining questions is how to obtain functionally-equivalent variants.

The ENACT IoT diversity-by-design tool takes as input a single deployment or behaviour specification and generates multiple diverse specifications. Within a DevOps context, it is important and necessary to keep the diversity generation fully

17. https://owaspsamm.org/model/implementation/secure-deployment/

automatic, instead of relying on developer's manual effort to diversify systems (such as the traditional N-Version Programming approach). Developers can focus on a single line of code to achieve frequent iteration, and the diversification tool, as part of automatic building step, will generate diversified versions automatically [14].

Automated diversity is a promising means of mitigating the consequences of a security breach. However, current automated diversity techniques operate on individual processes, leveraging mechanisms available at the lower levels of the software stack (in operating systems and compilers), yielding a limited amount of diversity. In this section, we present a novel approach for the automated synthesis of diversified protocols between processes. This approach builds on (i) abstraction, where the original protocol is modelled by a set of communicating state machines, (ii) automated synthesis, applying mutation operators onto those protocols, which produces semantically-equivalent, yet phenotypically-different protocols, and (iii) automated implementation of these protocols through code generation.

The tool is currently in an experimental stage. Automatic diversity of communication protocols is a novel technology, yet without convincing implementation and applications, to the best of our knowledge. Therefore, our focus is currently on the theoretical feasibility of the idea and the experimental evaluation of its effects. In the next step, we will improve the user experience of the tool and its applicability to practical scenarios.

Mass-produced software applications denote clonal applications, with thousands or millions of identical siblings. Think of, for example, a popular mobile application installed on millions of mobile phones, or software embedded into a widely-used connected device. To mitigate the risks of such large mono-cultures, diversity is typically automatically introduced either in a generic way, typically at the OS level, oblivious of the actual logic and semantics of the software, or in some very specific places, typically low-level libraries reused across applications, in order to improve security. This leaves most of the actual business logic unchanged, unaffected by the diversity. In addition, diversity often affects individual processes, but leaves the communication between processes intact.

A more holistic approach to diversity is challenging. Consider a typical client-server application, where multiple clients interact with a server, and where each client has a different implementation, and a different way of communicating with the server. This would significantly hinder a hacker, be it a human being or a machine, when attempting to generalize an attack through all possible protocols. This would make large-scale exploits a time-consuming and costly endeavour for hackers. Yet, the engineering, e.g., the production, maintenance and integration, of such levels of diversity raises several challenges. How to ensure that each implementation still behaves as specified? How to ensure that each client is still able to communicate with the server, without information loss or distortion? How to ensure that different clients are fundamentally (i.e., sufficiently) different, and not merely cosmetically different? How to keep the development and operation costs of a diversified system significantly lower than the cost of mitigating large scale attacks?

We have seen that abstraction, synthesis and automated implementation can yield a convincing solution to introduce a wide diversity into protocols, for example between a device and a gateway, or a web/mobile app and a server. This approach:

- abstracts protocols into (i) a structural view describing the messages to be exchanged, and (ii) a behavioral view based on state machines describing how those messages are exchanged between the participants, including sequencing and timing.
- combines and applies a number of atomic mutations to this protocol model, yielding a large number of diversified protocols, which operate differently, still with the same semantics.
- automatically implements protocols, diversified or not, by generating fully operational code targeting C, Go, Java and JavaScript, able to run on a wide range of platforms.

Our empirical assessment indicated that this approach implies a reasonable overhead in terms of execution time, memory consumption and bandwidth, fully compatible with the requirements of mass-produced software. We also showed that this approach could generate a significant amount of diversity. Our assumption was that this diversity would contribute to the diversity-stability hypothesis, i.e., this would make the whole ecosystem more robust by making it less likely for an exploit to propagate to the whole population. In other words, if the protocol between a specific client and the server could be observed, analysed and eventually understood, this would not systematically imply that all other diversified protocols could be understood following the very same procedure. In this section, we briefly describe the mechanisms and the corresponding tools we developed to automatically generate the diverse protocols. Technical details and the experiment results can be found in our conference paper [29].

Our approach relies on ThingML [26] for the specification of protocols. ThingML provides a way to formalize the messages involved in protocols, in a comparable way to what Protocol Buffer proposes. In addition, ThingML provides a mean to formalize the behavior of protocols through state machines. ThingML specifications are both human-readable and machine-readable, which makes it possible to analyse protocols at a high-level of abstraction and to fully automate the implementation of those protocols through code generation. In the next-subsection, we present relevant aspects of ThingML on our motivating example.

We model communication protocols as a set of communicating state-machines, encapsulated into components. A protocol typically involves two roles: (i) a client, i.e., a device, a web-browser or a mobile app, and (ii) a server, i.e., a gateway or a Cloud back-end. The clients and the server need to agree on a common API. Since communication is typically asynchronous in a distributed system, the common API is specified as a set of messages. Next, this API is imported by the client component and the server component, and the messages are organized into ports.

The ultimate goal of our approach is to diversify the wire image of protocols. Diversifying the wire image of protocols basically means shuffling the sequence of bytes exchanged over the network e.g., turning the payloads while ensuring the interoperability between the client and the server.

4.5 Conclusions

This chapter summarizes our effort in the ENACT project towards automatic software deployment for Smart IoT Systems. Automatic deployment is a cornerstone of DevOps, as it connects development with operation, and ensures that changes on the software will be placed into the production in a correct and prompt way.

Although there are already mature deployment solutions for Cloud computing in the market, automatic deployment for smart IoT systems is still an open problem. The main challenges are from two fundamental characters of smart IoT systems: First, an IoT application involves software running at all types of resources along the Cloud-Edge-IoT continuum, and it is difficult to provide a consistent way to support the deployment on all those different types of resources. Second, an IoT application in the production stage usually contains many subsystems of Edge and IoT devices, each of which serves a particular user or manages a particular part of the physical world. It is difficult to deploy a new change on the software to all those subsystems regardless of the different contexts and status among them.

During the ENACT project, we conducted research aiming at these two challenges, resulting in two ENACT enablers, namely GeneSIS and DivEnact. We briefly introduced how these enablers work, both as individual tools and as an integrated deployment bundle for the automatic deployment of SIS. More details about the theories, implementations and use cases can be founded in our recent publications [15, 44]. In this chapter, we focused on the mechanisms and practices of using these tools to ensure the trustworthiness of the deployment software, including the availability of software components on unstable resources, the deployment support of security and privacy mechanisms, and the automatic generation and maintenance of software diversity towards a more secure systems.

In the next step, we will extend the concepts and implementation of automatic deployment into the more general edge computing domain, providing an engineering solution for the core problem of edge computing, i.e., the distribution and offloading of computation among the complex and dynamic resources. Currently, the deployment is driven by manually define deployment models which embeds the resource allocation and the constraints about software-device mapping. An important future plan is to introduce intelligence into automatic deployment, which learns from historical deployments and their effects to automatically assign software parts to the proper resources.

References

[1] C. R. Aguayo Gonzalez, C. B. Dietrich, and J. H. Reed. "Understanding the software communications architecture". In: *IEEE Communications Magazine* 47.9 (2009), pp. 50–57.

[2] Carlos Ansótegui *et al.* "Satisfiability modulo theories: An efficient approach for the resource-constrained project scheduling problem". In: *Ninth Symposium of Abstraction, Reformulation, and Approximation*, 2011.

[3] Matej Arta *et al.* "Model-driven continuous deployment for quality devops". In: *Proceedings of the 2nd International Workshop on Quality-Aware DevOps*. 2016, pp. 40–41.

[4] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. 3rd. Addison-Wesley Professional, 2012. ISBN: 0321815734.

[5] Benoit Baudry and Martin Monperrus. "The Multiple Facets of Software Diversity: Recent Developments in Year 2000 and Beyond". In: *ACM Comput. Surv.* 48.1 (Sept. 2015). ISSN: 0360-0300. DOI: 10.1145/2807593. URL: https://doi.org/10.1145/2807593.

[6] Gordon S. Blair, Nelly Bencomo, and Robert B. France. "Models@run.time". In: *IEEE Computer* 42.10 (2009), pp. 22–27.

[7] Miquel Bofill *et al.* "Solving constraint satisfaction problems with SAT modulo theories". In: *Constraints* 17.3 (2012), pp. 273–303.

[8] Maria Paola Bonacina, Stéphane Graham-Lengrand, and Natarajan Shankar. "Satisfiability modulo theories and assignments". In: *International Conference on Automated Deduction*. Springer. 2017, pp. 42–59.

[9] Antonio Bucchiarone, Antonio Cicchetti, and Annapaola Marconi. "Exploiting multi-level modelling for designing and deploying gameful systems". In: *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE. 2019, pp. 34–44.

[10] Sylvain Cherrier *et al.* "D-lite: Distributed logic for internet of things services". In: *2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*. IEEE. 2011, pp. 16–24.

[11] F. Cirillo *et al.* "A Standard-Based Open Source IoT Platform: FIWARE. In: *IEEE Internet of Things Magazine* 2.3 (2019), pp. 12–18. DOI: 10.1109/IOTM.0001.1800022.

[12] Benoit Combemale and Manuel Wimmer. "Towards a Model-Based DevOps for Cyber-Physical Systems". In: *Software Engineering Aspects of Continuous Development*, 2019.

[13] Rustem Dautov and Hui Song. "Towards Agile Management of Containerised Software at the Edge". In: *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*. Vol. 1. IEEE. 2020, pp. 263–268.

[14] Rustem Dautov and Hui Song. "Towards IoT Diversity via Automated Fleet Management". In: *MDE4IoT/ModComp@ MoDELS*. 2019, pp. 47–54.

[15] Rustem Dautov, Hui Song, and Nicolas Ferry. "A Light-Weight Approach to Software Assignment at the Edge". In: *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*. IEEE. 2020, pp. 380–385.

[16] Johannes Eder *et al.* "Bringing DSE to life: exploring the design space of an industrial automotive use case". In: *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE. 2017, pp. 270–280.

[17] Johannes Eder *et al.* "From deployment to platform exploration: automatic synthesis of distributed automotive hardware architectures". In: *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. 2018, pp. 438–446.

[18] Nicolas Ferry and Phu H. Nguyen. "Towards Model-Based Continuous Deployment of Secure IoT Systems". In: *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 2019, pp. 613–618.

[19] Nicolas Ferry *et al.* "CloudMF: Model-Driven Management of Multi-Cloud Applications". In: *ACM Transactions on Internet Technology (TOIT)* 18.2 (2018), p. 16.

[20] Nicolas Ferry *et al.* "Continuous Deployment of Trustworthy Smart IoT Systems". In: *The Journal of Object Technology* (2020).

[21] Nicolas Ferry *et al.* "Genesis: Continuous orchestration and deployment of smart IoT systems". In: *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. IEEE. 2019, pp. 870–875.

[22] M. Frustaci *et al.* "Evaluating Critical Security Issues of the IoT World: Present and Future Challenges". In: *IEEE Internet of Things Journal* 5.4 (2018), pp. 2483–2495. DOI: 10.1109/JIOT.2017.2767291.

[23] Anne Gallon *et al.* "Making the Internet of Things More Reliable Thanks to Dynamic Access Control". In: *Security and Privacy in the Internet of Things: Challenges and Solutions* 27 (2020), p. 61.

[24] Nam Ky Giang *et al.* "Developing IoT applications in the fog: a distributed dataflow approach". In: *Internet of Things (IoT), 2015 5th International Conference on the*. IEEE. 2015, pp. 155–162.

[25] Object Management Group. "Deployment and Configuration of Component-based Distributed Applications Specification". In: *OMG Available Specification Version 4.0 formal/06-04-02* (2006).

[26] Nicolas Harrand *et al.* "ThingML: A Language and Code Generation Framework for Heterogeneous Targets". In: *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*. MODELS 16. Saint-malo, France: Association for Computing Machinery, 2016, pp. 125–135. ISBN: 9781450343213.

[27] Stéphane Lavirotte *et al.* "A generic service oriented software platform to design ambient intelligent systems". In: *Proceedings of the 2015 ACM International Conference on Pervasive and Ubiquitous Computing*. ACM. 2015, pp. 281–284.

[28] Amardeep Mehta *et al.* "Calvin Constrained-A Framework for IoT Applications in Heterogeneous Environments". In: *37th International Conference on Distributed Computing Systems*. IEEE. 2017, pp. 1063–1073.

[29] Brice Morin *et al.* "Engineering software diversity: A model-based approach to systematically diversify communications". In: *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. 2018, pp. 155–165.

[30] Håvard Myrbakken and Ricardo Colomo-Palacios. "DevSecOps: A Multivocal Literature Review". In: *Software Process Improvement and Capability Determination*. Ed. by Antonia Mas *et al.* Cham: Springer International Publishing, 2017, pp. 17–29. ISBN: 978-3-319-67383-7.

[31] NESSI. *Cyber physical systems: Opportunities and challenges for soft- ware, services, cloud and data*. NESSI White paper. 2015.

[32] P. H. Nguyen *et al.* "SoSPa: A system of Security design Patterns for Systematically engineering secure systems". In: *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 2015, pp. 246–255. DOI: 10.1109/MODELS.2015.7338255.

[33] Phu H. Nguyen, Phu H. Phung, and Hong-Linh Truong. "A Security Policy Enforcement Framework for Controlling IoT Tenant Applications in the Edge". In: *Proceedings of the 8th International Conference on the Internet of Things*, IOT 18. Santa Barbara, California, USA: ACM, 2018. ISBN: 9781450365642.

[34] Phu H. Nguyen *et al.* "Advances in deployment and orchestration approaches for IoT – A systematic review". In: *2019 IEEE International Congress On Internet of Things (ICIOT)*. Milan, Italy: IEEE, 2019, pp. 53–60.

[35] Phu H. Nguyen *et al.* “An extensive systematic review on the Model-Driven Development of Secure Systems”. In: *Information and Software Technology* 68 (2015), pp. 62–81. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2015.08.006. URL: http://www.sciencedirect.com/science/article/pii/S0950584915001482.

[36] Adrian Noguero, Angel Rego, and Stefan Schuster. “Towards a Smart Applications Development Framework”. In: *Social Media and Publicity* 27 (2014). URL: https://bitbucket.org/jasonjxm/smool,%202011-2020.

[37] OMG. *Deployment and Configuration of Component-based Distributed Applications Specification, v4.0*. Tech. rep. Object Management Group, Inc., 2006. URL: https://www.omg.org/spec/DEPL/4.0/PDF.

[38] Temel Öncan. “A survey of the generalized assignment problem and its applications”. In: *INFOR: Information Systems and Operational Research* 45.3 (2007), pp. 123–141.

[39] Allen Parrish, Brandon Dixon, and David Cordes. “A conceptual foundation for component-based software deployment”. In: *Journal of Systems and Software* 57.3 (2001), pp. 193–200. ISSN: 0164-1212.

[40] David W Pentico. “Assignment problems: A golden anniversary survey”. In: *European Journal of Operational Research* 176.2 (2007), pp. 774–793.

[41] Subhav Pradhan *et al.* “Chariot: Goal-driven orchestration middleware for resilient IoT systems”. In: *ACM Transactions on Cyber-Physical Systems* 2.3 (2018), pp. 1–37.

[42] Davide Di Ruscio, Richard F. Paige, and Alfonso Pierantonio, eds. *Special issue on Success Stories in Model Driven Engineering*. Vol. 89, Part B. Elsevier, 2014.

[43] Ana C Franco da Silva *et al.* “OpenTOSCA for IoT: automating the deployment of IoT applications based on the mosquitto message broker”. In: *Proceedings of the 6th International Conference on the Internet of Things*. ACM. 2016, pp. 181–182.

[44] Hui Song *et al.* “Model-based fleet deployment of edge computing applications”. In: *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. 2020, pp. 132–142.

[45] Hui Song *et al.* “On architectural diversity of dynamic adaptive systems”. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 2. IEEE. 2015, pp. 595–598.

[46] Michael Vögler *et al.* “A scalable framework for provisioning large-scale IoT deployments”. In: *ACM Transactions on Internet Technology (TOIT)* 16.2 (2016) pp. 1–20.

[47] Changsheng You *et al.* “Energy-efficient resource allocation for mobile-edge computation offloading”. In: *IEEE Transactions on Wireless Communications* 16.3 (2016) 1397–1411.

DOI: 10.1561/9781680838251.ch5

Chapter 5

A DevOps Toolchain for Managing Actuation Conflicts in Smart IoT Systems

By Gérald Rocher, Thibaut Gonnin, Franck Dechavanne, Stéphane Lavirotte and Jean-Yves Tigli

5.1 Introduction

To assist users in their daily lives, Smart IoT Systems (SIS) have long been limited to the exploitation of environmental information; the use of 'smart objects' was mainly motivated by their ability to collect these information from sensors. However, in many areas such as home automation, factory 4.0, Intelligent Transportation Systems (ITS), etc., SIS are no longer limited to collecting sensor data to infer actionable information, they also interact with the physical environment through actuators. This evolution brings new challenges that the scientific community has to meet in collaboration with industrial players, as evidenced by the numerous calls from the European Commission, in which the ENACT project is part of: *"Most of the today's IoT systems are however mainly focused on sensors, whereas in the future actuation and smart behaviour will be the key points. Platforms should provide connectivity and intelligence, actuation and control features"* [9], p. 100.

5.1.1 SIS Actuation Challenges

By physically interacting with their environment through actuators, SIS become critical; in the absence of end-to-end human control (utopian in complex operational contexts such as ITS, etc.), they have immediate impacts on the physical environment with all the social and economical risks that this may entail. At design-time, these applications must be conceived to formally prevent any undesirable effect in the physical environment, whether caused by sending contradictory or simultaneous commands to the actuators. At run-time, it is yet necessary to ensure that the effects produced in the physical environment are in line with the expectations. Indeed, the lack of a perfect model of the physical environment (of a complex nature) prevents designers from fully predicting the effects of the commands sent to the actuators; effects that are therefore likely to be hampered by possibly disruptive surrounding physical processes. These challenges bring with them heightened concerns about *trustworthiness* of SIS which includes, among others, *safety* and *reliability* aspects. As defined in [13], safety concerns are related to the ability of SIS to prevent catastrophic consequences for humans and the physical environment; reliability concerns are related to the ability of SIS to deliver predictable performance in expected conditions. While the DevOps methodology is part of the good practices in software development and is applicable to SIS limited to merely collect sensor data, it requires new tools in the realm of trustworthy SIS, both at Dev and Ops Times.

5.1.2 DevOps Still Lacks the Tools to Meet These Challenges

While the DevOps approach is not specific to a particular field of applications, many challenges arise when it comes to applying it to the field of SIS. DevOps practices are therefore still far from being fully adopted in their development, notably due to a lack of key enabling tools [1, 34]. Among these key tools, those capable of taking into account SIS operating in open and complex environments and requiring continuous testing at run-time (i.e., in-situ) beyond the tests traditionally conducted on emulated and simulated infrastructures, are missing [1]. In general, there is a lack of tools that address the trustworthiness of SIS, which, as far as this chapter is concerned, is about safety and reliability aspects, exacerbated by the ability of SIS to act in the physical environment through actuators. A study of the literature around the actuation problem [19] shows that, without even considering the DevOps approach, this problem is still in its infancy in the IoT field and therefore still open. Moreover, the approaches proposed in the literature for managing actuation in SIS are often (1) monolithic, they do not or hardly meet the best practices advocated by the DevOps approach; (2) applied to controlled operational environments; they have a software vision of the problem by focusing on the commands sent to the actuators more than on the effects they produce [29].

The work carried out as part of the ENACT European project and described throughout this chapter aims to fill this gap by proposing a toolchain meant to be integrated into the DevOps framework and meeting end-to-end SIS actuation challenges.

5.1.3 An End-to-end DevOps Toolchain

The contribution is built around two complementary toolsets, deployed throughout the DevOps life-cycle phases. This combination of tools aims to improve the trustworthiness of SIS within a framework which, as advocated by the DevOps approach, creates synergies and foster communication between development and operations activities.

At design-time (Dev), a complete toolset is developed for identifying and locally resolving actuation conflicts [12], i.e., (a) preventing contradictory or simultaneous commands to be sent to actuators (direct conflicts), (b) preventing, as much as it can be, antagonistic effects to occur in the physical environment (indirect conflicts). At run-time (Ops), a first tool observes specific environmental features and quantitatively assesses the effectiveness of the SIS, i.e., for the extent to which it produces the expected effects. A second tool is meant to analyse drifts in effectiveness. It makes clear the symptoms of the drifts in effectiveness, providing guidance to designers to help them investigate their possible root causes. Within the framework of the ENACT project, other investigation tools may be used to complement this latter tool, such as, for instance, the Root Cause Analysis (RCA) (Section 8.3) toolset. The analysis resulting from these tools then trigger a new design phase, closing the DevOps life-cycle loop.

The rest of the chapter is organized as follows. Section 5.2 describes the SIS actuation management toolset along with the workflow (Section 5.2.2) involved in the identification and resolution of direct and indirect actuation conflicts. Section 5.3 describes the behavioural drift assessment (Section 5.3.2) and analysis (Section 5.3.3) toolset. Section 5.4 demonstrates, throughout a smart-home use-case described in Section 5.4.1, the complementarity and relevancy of both toolset as part of the DevOps life-cycle. This use-case involves two consecutive DevOps cycles to converge towards a SIS with satisfactory behaviour.

5.2 Overview of the SIS Actuation Conflict Management Toolset

IoT devices, at the edge of SIS infrastructures, have long been leveraged for their capacity at gathering environmental data from sensors, paving the way for decision making support systems covering a broad range of application domains from smart-health, smart-city to smart-grid, Factory 4.0, etc. just to name a few. However, beyond merely gathering data from sensors, new challenges arise as soon as it comes to leverage IoT devices to interact with the physical environment through actuators that turn commands received from SIS into physical effects. One of these challenges concerns the actuation conflict management. Such conflicts are likely to occur when different applications compete for accessing (1) shared actuators at the edge of the IoT infrastructure (direct conflicts), e.g., simultaneously applying ON and OFF to a light bulb, and/or (2) shared physical properties (indirect conflicts), i.e., turning ON both a cooler and a heater in the same room.

The actuation conflict management challenge goes beyond the technological challenge that shared multi-layered IoT infrastructures usually meet by, for instance, providing sensors with access control mechanisms. Indeed, besides this technological challenge, actuation conflict management also poses a semantical challenge; accounting for the locality of the actuators and the physical properties they act on is here essential. In the realm of trustworthy SIS, actuation conflict management is of paramount importance. Designers must prevent SIS from producing any undesirable effects in the physical environment, whether it is caused by sending contradictory or simultaneous commands to the actuators. To this end, designers must be provided with decision support tools that can assist them in identifying and resolving direct and indirect actuation conflicts, and in deploying relevant, yet robust and safe Actuation Conflict Managers (ACM).

In the sequel, a complete toolset for identifying and resolving actuation conflicts is introduced. This toolset consists of three stages throughout the DevOps life-cycle, as described in Fig. 5.1).

Underlying the tools for actuation conflict identification and resolution, a meta-model, denoted Workflow and Interaction Model for Actuation Conflict management (WIMAC), is used to build a structural model of the SIS upon deployment, implementation and physical environment models (phase 1 in Fig. 5.1) [27]. WIMAC provides a modelling language for describing inter-relationships between SIS software components and actuators at the edge of the infrastructure along with their effects on the physical environment.

On the basis of the structural model, potential direct and indirect actuation conflicts are identified (phase 2 in Fig. 5.1). Actuation conflict identification is based on Attributed Graph Grammar (AGG) rules [18] meant to detect conflicting patterns; actuation conflict resolution is based on AGG rewriting rules meant to instantiate local ACMs. DevOps approach aims to provide continuous and rapid software deployment capabilities. In accordance with this approach, a set of pre-configured off-the-shelf and ready-to-use ACMs are offered to designers.

Third, besides off-the-shelf ACMs, a complete formal verification flow is proposed in the Discrete EVent system Specification formalism (DEVS) [39] for

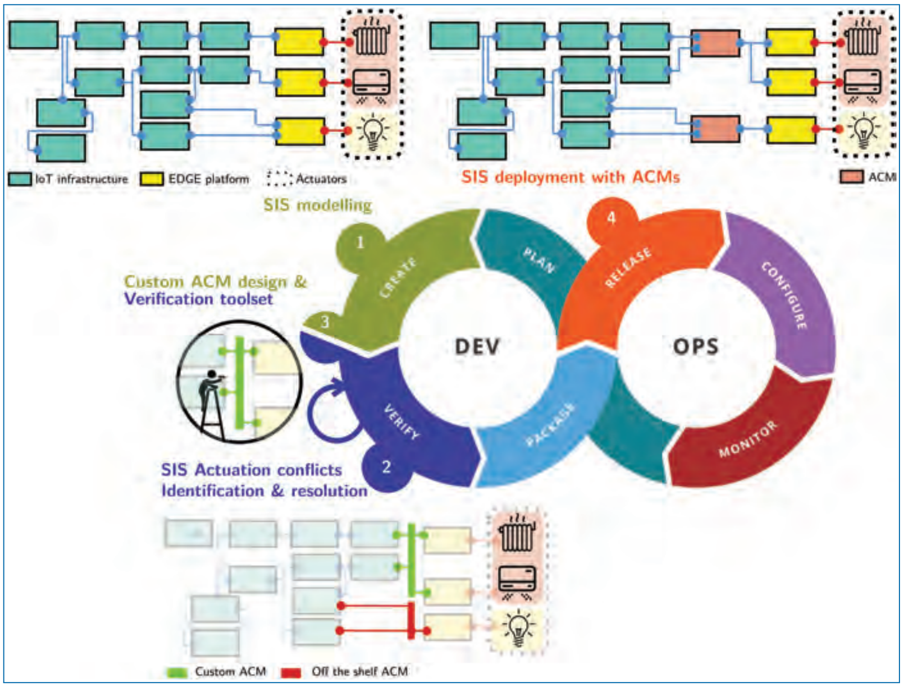


Figure 5.1. Usage of Actuation Conflict Management (ACM) tool-set throughout the DevOps life-cycle.

designing and verifying reusable custom ACMs, covering different implementation strategies at the edge of the infrastructure (phase 3 in Fig. 5.1).

Finally, the WIMAC-based structural model, with ACMs instantiated, is transformed back to deployment and implementation models (phase 4 in Fig. 5.1).

5.2.1 Beyond the State of the Art

The problem of identifying and resolving IoT-based systems actuation conflicts is still in its infancy. While most of the existing solutions focus on the identification and the resolution of direct actuation conflicts (a legacy of the technical challenge mentioned above), few focus on identifying and resolving indirect ones [19, 21]. Moreover, the solutions proposed in the literature raise two main problems when it comes to applying them to the DevOps framework; (1) most of them require an a priori knowledge on the system components and the rules governing their evolution [33]; (2) the solutions proposed are thereby monolithic, they implement global identification and resolution mechanisms not easily reusable (e.g., [2, 21, 22]).

In [24], authors recognize that interactions between IoT devices are an increasing cause of safety and security violations whose detection *"[...] requires a holistic view of installed apps, component devices, their configurations, and more importantly,* *how they interact*". This is the approach followed by the SIS actuation conflict management toolset presented in the sequel.

5.2.2 Actuation Conflict Management Workflow

The actuation conflict management workflow is depicted in Fig. 5.2.

DevOps approach provides designers with deployment and implementation models from which it is possible to extract the structural interactions between the SIS software components down to the devices and the actuators they embed. To identify indirect conflicts, however, it is necessary to describe the effects that the actuators produce in the environment in which they operate. Unless each actuator is accompanied by semantic annotations, these descriptions have to be provided by the designers, using a model of the physical environment.

From these models, an holistic description model is generated. This model is built on a metamodel called WIMAC (Workflow and Interaction Model for Actuation Conflict management). This metamodel (Fig. 5.3) provides a language to describe (1) the inter-relationships between software components, down to the

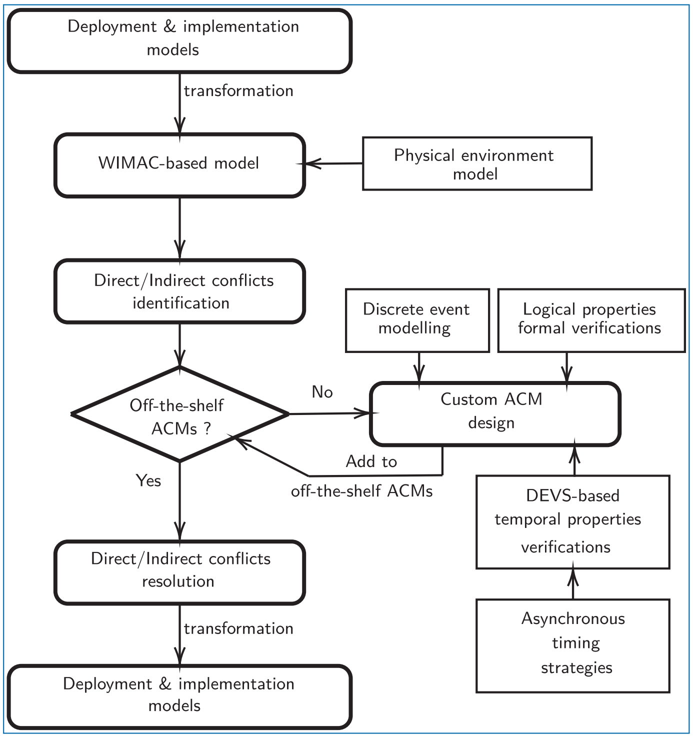

Deployment & implementation models
transformation
WIMAC-based model
Physical environment model
Direct/Indirect conflicts identification
Discrete event modelling
Logical properties formal verifications
Off-the-shelf ACMs ?
No
Custom ACM design
Add to off-the-shelf ACMs
Yes
DEVS-based temporal properties verifications
Direct/Indirect conflicts resolution
transformation
Asynchronous timing strategies
Deployment & implementation models

Figure 5.2. Actuation conflicts identification and resolution workflow.

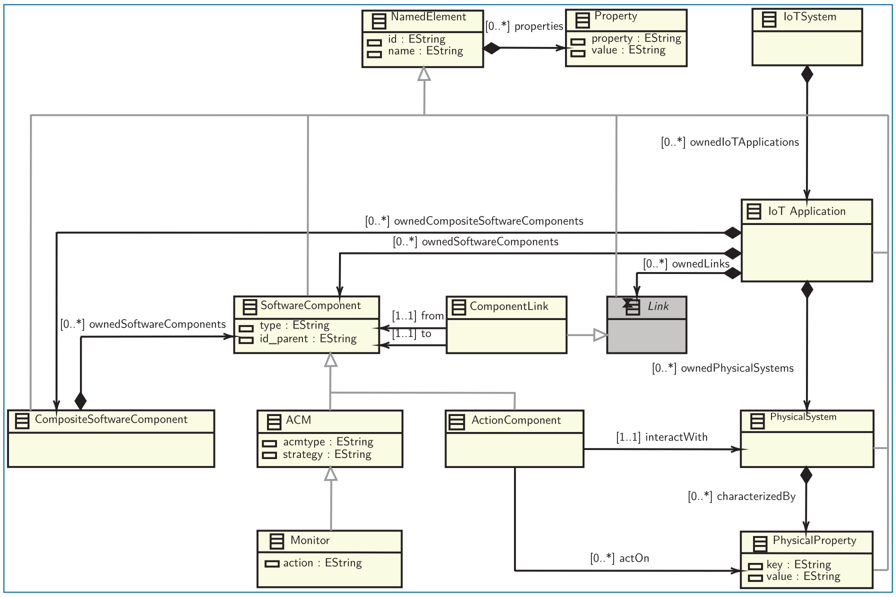


Figure 5.3. WIMAC meta-model.

actuators at the edge of the infrastructure as well as (2), the effects that the actuators produce in the physical environment in which they operate. The WIMAC metamodel main entities are the following:

1. **SoftwareComponents** are black-box components. They conceptualize single applications or composite applications described through implementation models (e.g., Node-RED[1] flow). The actuation conflict management solution proposed in the sequel is based solely on the structural links that exist between software components for inferring their interrelationships, identifying and resolving potential conflicts.
2. **ActionComponents** are SoftwareComponents controlling actuators,
3. **PhysicalSystems** are spatially delimited physical entities whose properties can be changed by ActionComponents (e.g. the temperature in the kitchen).

Relying on the WIMAC-based model, potential direct and indirect actuation conflict points can be identified and monitoring ACMs instantiated locally. This first step is automatically achieved through a set of predefined Attributed Graph Grammar (AGG) rules [18] in the form of attributed graphs. These rules define (1) conflict patterns to be identified in the WIMAC-based model and (2) associated

1. https://nodered.org

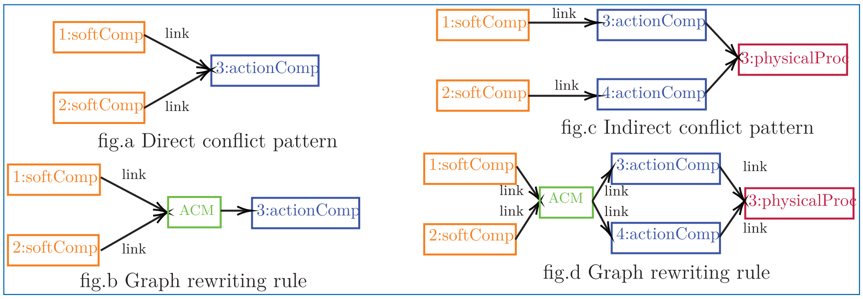


Figure 5.4. Example of AGG identification and rewriting rules for direct (left) and indirect (right) actuation conflicts.

graph transformations to be applied on the model so as to instantiate monitoring ACMs (Fig. 5.4).

DevOps approach aims to provide continuous and rapid deployment capabilities. To support this objective, a set of off-the-shelf generic ACMs are proposed to designers thereby, replacing dummy ACMs with relevant concrete ones. Although this approach requires designers to take responsibility on the ACMs to be instantiated before applications are deployed, it has several advantages. (1) it helps reinforcing SIS trustworthiness, specifically as actuation conflict management implies some semantic background humans are better able to grasp considering SIS complexity globally; (2) this is all the more important as ACMs are instantiated locally, addressing points of potential conflict in the design. The advantage here is that software components are regarded as black boxes, ACMs do not change their logic.

While off-the-shelf reusable ACMs are relevant for resolving common actuation conflicts, they may not be suitable for some particular cases. To address this concern, state-of-the-art Model Driven Engineering (MDE) tools are leveraged, allowing designers to develop custom, yet robust and safe ACMs, throughout a two-level design workflow (Fig. 5.5).

At the first level, logic of the custom ACMs are defined through Finite State Machines (FSM) built on the basis Event-Condition-Action rules. These conceptual models allow logical properties (e.g. completeness, safety, liveness, etc.) to be formally verified using state-of-the-art methodologies [3].

At the second level, implementation models allow temporal properties to be formally verified by applying different asynchronous timing strategies. Implementation models are proposed to be described through the DEVS formalism (Discrete EVent system Specification)[39]. This formalism brings key advantages in the realm of trustworthy SIS and DevOps.

1. DEVS atomic models allow to encapsulate conceptual models into DEVS Atomic models, coupled with synchronizers that can implement different

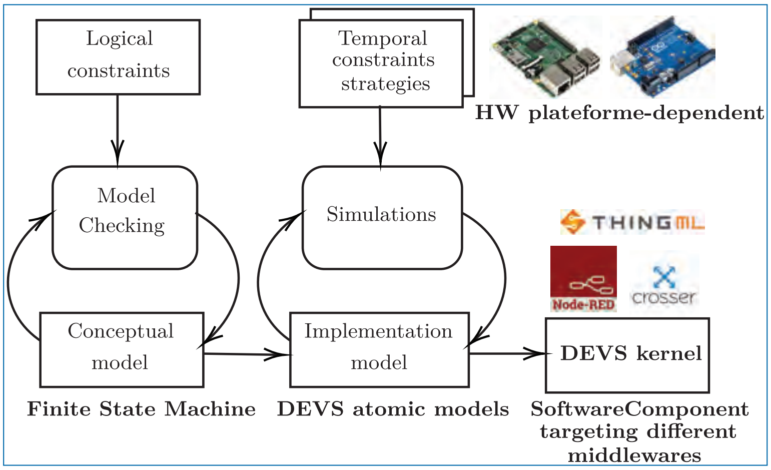


Figure 5.5. Custom ACM design workflow.

timing strategies on the inputs of the FSM and targeting different hardware platforms. This approach is particularly relevant considering that SIS software components are likely to be deployed on resource-constrained platforms at the edge of the IoT infrastructure, governed by asynchronous events relative to wireless communication protocols, computational capabilities, etc. [35]. Several modelling and simulation tools are available to facilitate this process [6, 25, 36],

2. It provides a common representation to different discrete event modelling formalisms (including Petri Nets, FSM, etc.) [41]. Designers are therefore not bounded to a particular modelling framework when designing custom ACMs.
3. DEVS Atomic models, once verified for temporal properties, can be translated into any high-level programming language (e.g., C, C++, C#, etc.) and compiled for further deployment. This allows to build a library of reusable off-the-shelf DEVS-based ACM software components (a.k.a., DEVS kernels) targeting different implementation strategies (i.e., hardware platforms).

5.3 Overview of the SIS Behaviour Monitoring and Analysis Toolset

While, during the development phase, the ACM tool allows for the identification, analysis and resolution of actuation conflicts, it remains, however, based on the global architecture of the SIS and its operational environment, i.e., and is therefore based on a priori acquired knowledge. However, by interacting with the physical environment through actuators, SIS inherits the *complex* nature of this open environment. Thus, assuming that the knowledge acquired a priori is complete and immutable is illusory, no matter how complicated the associated models may be; an infinite number of unexpected physical processes are likely to disrupt, at any time, operation of the SIS. In the realm of trustworthy SIS, the ACM tool alone thus cannot meet the concerns of reliability and safety. It is therefore necessary to complement this tool with a systemic approach, required to the modelling of complex SIS [30].

Without being able to predict the behaviour of SIS in their design phase, a DevOps tool-set is developed for analysing SIS behavioural drifts at run-time (Fig. 5.6). The aim of the first tool described in the sequel is to quantitatively assess SIS *effectiveness* at run-time, i.e., the extent to which the effects produced in the physical environment are legitimate (phase 2 in Fig. 5.6). It is no longer a question of describing the global architecture of the SIS, but of describing a

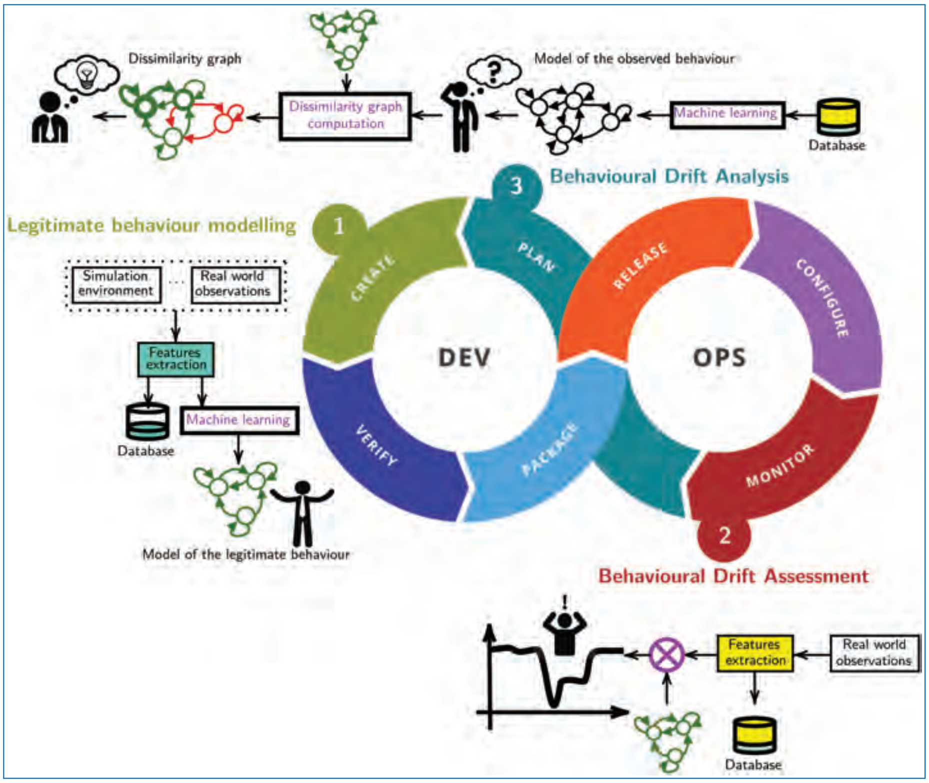


Figure 5.6. Usage of the Behavioural Drift Analysis (BDA) tool-set throughout the DevOps life-cycle.

model of the effects they have to legitimately produce in the physical environment in different contexts (phase 1 in Fig. 5.6). However, being quantitative in nature, the effectiveness assessment is difficult to interpret by humans and does not allow for creating synergies and fostering communication between development and operations activities, as advocated by the DevOps approach. Indeed, from the designers' point of view, while the effectiveness assessment allows the detection of unexpected and/or abnormal behaviour at run-time, it does not allow their causes to be identified. Thereby, this hampers a new DevOps cycle from being initiated rapidly so as to implement the corrective actions required. To address this concern, a second tool is presented in the sequel. This tool (phase 3 in Fig. 5.6), produces a model of the behaviour observed in the field and compares this model with the model of the legitimate behaviour produced in phase 1. This results in a dissimilarity graph between both models which, without identifying the causes of the drifts in effectiveness, sheds light on the symptoms likely to explain them [29].

5.3.1 Beyond the State of the Art

The problem raised in this research, and the solution described in the sequel, concerns the detection, identification and intelligible representation of the symptoms characterizing SIS abnormal behaviours, for the purpose of empirical analysis. A recent systematic literature review carried out on the anomaly detection, analysis and prediction techniques in IoT environments [10], corroborates fairly well the novelty of the approach proposed in this work and the impacts it can have in the SIS community. Indeed, authors found gaps in the visualization of anomalies in IoT-based systems. They conclude that new methods and approaches are needed to represent intuitively IoT-based systems for analytical purposes. This is what the solution described in the sequel is all about.

5.3.2 Behavioural Drift Assessment Tool

Current software engineering approaches, which include formal testing and verification methodologies, claim for predictability. However, *"in practice, analytical modelling is increasingly proving inadequate, whenever it is agreed that one is not sure that something cannot be forgotten (the hypothesis of closing the model), that objective evidence is only evident in a given ideology (…), in other words, whenever one has to make the assumption that the phenomenon modelled is not complicated but complex"* [20] p. 19.

SIS, by their interaction with the open physical environment, are complex systems; At best, can we hope to get as close as possible to the desired behaviour without reaching it exactly: *"[...] as soon as a system is open there is no optimum and any equilibrium is in interaction with its environment"* [31]. As a result, while the formal testing and verification methods still need to be conducted at design-time, they are not sufficient; the effectiveness of the SIS, i.e., the extent to which SIS behave as expected, must also be continuously evaluated at run-time.

Analytical modelling being inadequate, SIS effectiveness assessment has to rely on a systemic model of the physical effects they are legitimate to produce in the physical environment [30]. In the sequel, it is proposed to build this model in the framework of the Input/Output Hidden Markov Models (IOHMM) [4]. This framework is broadly used in behavioural modelling approaches [14, 40] where it has many advantages [37]; (1) it is an *explainable graphical model* [15] part of the Dynamic Bayesian Networks (DBN) family; (2) it formalizes *conditional dependencies* between the effects and their stimuli (i.e., *contextual input*, *events*); (3) it handles *tolerances on expectations* that may reflect randomness of some expected behaviours (through probability theory) or epistemic gaps in SIS knowledge (through the possibility theory and the Transferable Belief Model (TBM), an extension of the Dempster-Shafer evidence theory).

Formally, an IOHMM is defined by the tuple $< Q, \vec{\pi}, A, \vec{B} >$ where:

- $Q = \{x_1, x_2, \ldots, x_N\}$, $N \in \mathbb{N}$, is the finite set of hidden states where $x_{(k)}$ denotes the hidden state at time $k \in \mathbb{N}$,
- $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)^T$ is the initial state distribution vector where π_i denotes the likelihood of the state i to be the first state of a state sequence,[2]
- A is the $N \times N$ state transition matrix, where each element a_{ij} of the matrix is an n-dimensional input distribution ($1 \leq i, j \leq N$). Thus, $a_{ij}(\vec{u}) = p(x_{(k+1)} = j | x_{(k)} = i, \vec{u}_{(k)} = \vec{u})$ denotes the likelihood of transitioning to state $x_{(k+1)} = j$ at time $k + 1$, given the current state $x_{(k)} = i$ and the contextual input vector $\vec{u}_{(k)} = \vec{u} \in \mathbb{R}^n$ at time k. The function p has here to be understood in the general framework of the uncertainty theory. While most of the hidden Markov-based models are defined in the probabilistic framework (i.e., p is a measure of probability), the model is also compatible with the possibility theory [28], the imprecise probabilities [11], etc.
- $\vec{B} = (b_1, b_2, \ldots, b_N)^T$ is the state emission vector, where each element b_i ($1 \leq i \leq N$) is an m-dimensional output distribution. $b_i(\vec{y}) = p(\vec{y}_{(k)} = \vec{y} | x_{(k)} = i)$ denotes the likelihood of observing the output vector $\vec{y}_{(k)} = \vec{y} \in \mathbb{R}^m$ at time k, i.e., the physical effects produced, while being in the state $x_{(k)} = i$. The output observation $\vec{y}_{(k)}$ at time k only depends on the state $x_{(k)}$ at time k.

2. In this work, we assume that the elements π_i of $\vec{\pi}$ are equally probable, i.e., equal to $\frac{1}{N}$.

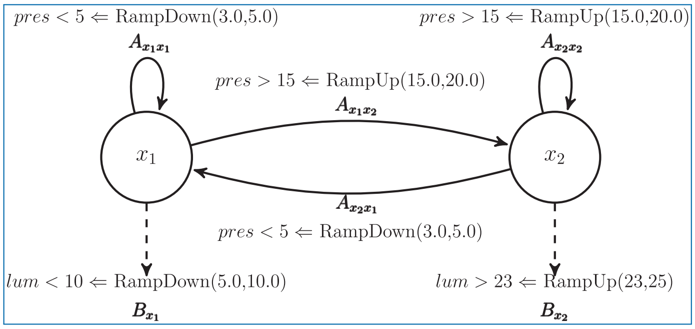

$pres < 5 \Leftarrow$ RampDown(3.0,5.0)
$A_{x_1x_1}$
$pres > 15 \Leftarrow$ RampUp(15.0,20.0)
$A_{x_2x_2}$
$pres > 15 \Leftarrow$ RampUp(15.0,20.0)
$A_{x_1x_2}$
x_1
x_2
$A_{x_2x_1}$
$pres < 5 \Leftarrow$ RampDown(3.0,5.0)
$lum < 10 \Leftarrow$ RampDown(5.0,10.0)
B_{x_1}
$lum > 23 \Leftarrow$ RampUp(23,25)
B_{x_2}

Figure 5.7. Possibilistic IOHMM model describing the legitimate effects that a luminosity control system must produce.

This model serves as a basis for efficient solutions to several inference problems [26]. Among these problems, the problem of inferring the likelihood of an observation sequence (i.e., $p((\vec{u}_{(k)}, \vec{y}_{(k)})_{k=1}^{K}))$ to have been generated by the model (filtering), is particularly close to the one of assessing SIS effectiveness whose solution is given by the forward algorithm, here below in its (qualitative) possibilistic form where the function p is a measure of possibility denoted by Π:

1. **Initialization** – $\forall x \in Q$:

$$\alpha_{(1)}(x) = \min\left(\pi_x, \Pi(\vec{y}_{(1)}|x)\right) \tag{5.1}$$

2. **Induction** – $\forall x, x' \in Q, \forall 2 \leq k \leq K$:

$$\alpha_{(k)}(x) = \min\left(\max_{x' \in Q}\left(\min\left[\alpha_{(k-1)}(x'), \Pi(x|x', \vec{u}_{(k-1)})\right]\right), \Pi(\vec{y}_{(k)}|x)\right) \tag{5.2}$$

3. **Termination**

$$\Pi\left((\vec{u}_{(k)}, \vec{y}_{(k)})_{k=1}^{K}\right) = \max_{x \in Q}(\alpha_{(K)}(x)) \tag{5.3}$$

An example is depicted in Fig. 5.7. The model describes the legitimate effects that a luminosity control system must produce where tolerances on expectations are described through distributions of possibility as depicted in Fig. 5.8. Here, $\vec{u}$ corresponds to the value of a presence sensor, $\vec{y}$ corresponds to the value of a luminosity sensor.

5.3.3 Behavioural Drift Analysis Tool

On the basis of the IOHMM model and field observations, the effectiveness assessment allows the detection of behavioural drifts, consequence of unexpected

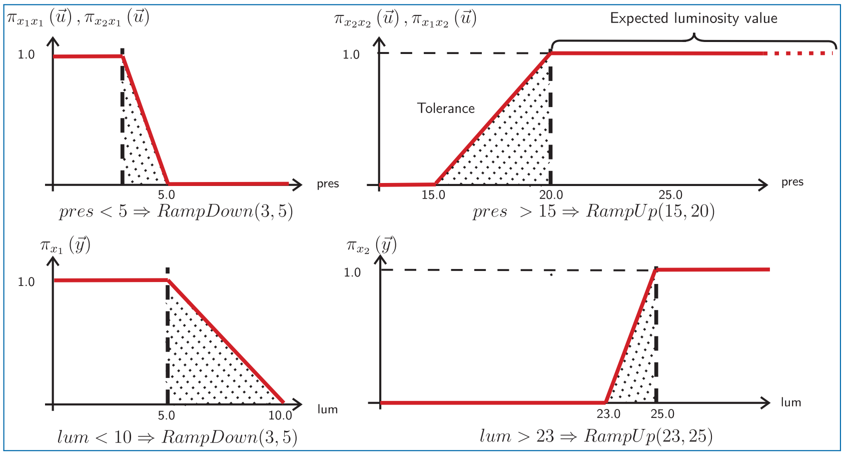


Figure 5.8. Distributions of possibility defining tolerances on expectations of the model depicted in Fig. 5.7.

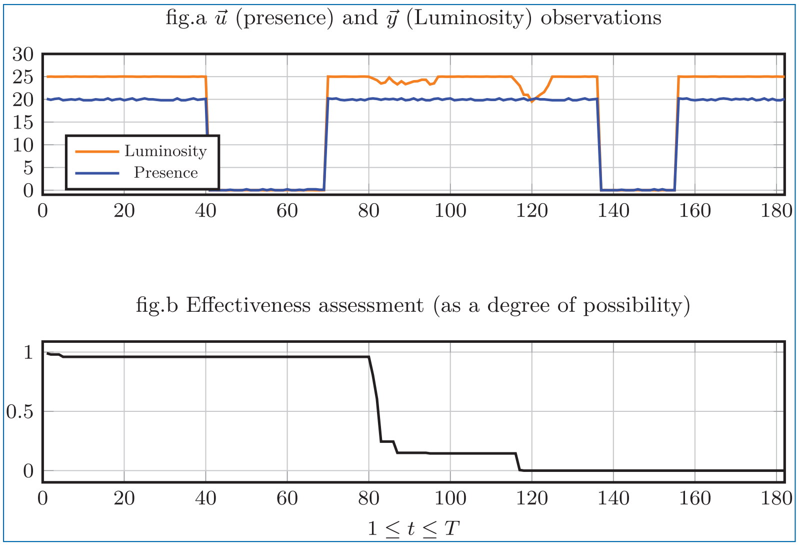


Figure 5.9. Effectiveness assessment results obtained on the model depicted in Fig. 5.7 from synthetic observations applied to the forward algorithm.

behaviours, whether legitimate or not. It complements the formal testing and verification approaches carried out at design-time. However, while the model underlying this evaluation is explainable, the same does not apply to the evaluation itself. Indeed, as a quantitative evaluation, it does not provide designers with information that would support them in understanding the reasons for its drifts, limiting de facto their ability to quickly take corrective actions and mitigate risks. The tool described in the sequel is meant to fill this gap.

This tool is mainly based on a generic clustering-based algorithm meant to learn IOHMM structure (states and state transitions) and parameters (distributions) from field observations. This algorithm consists in segmenting $\vec{u}$ and $\vec{y}$ observation spaces into a finite number of relevant regions such that each region represents respectively an input context and a discrete state, i.e., it is assumed that there exists a bijection between state and observation spaces [32], thereby taking advantage of understanding the structure of the IOHMM from the observation space [16].

Considering the observation sequence $(\vec{u}_{(k)}, \vec{y}_{(k)})_{k=1}^{K}$, the algorithm works as follows (detailed in [29]):

1. Get the set $\mathcal{Y}$ of clusters from $(\vec{y}_{(k)})_{k=1}^{K}$ such that each observation $\vec{y}_{(k)}$ is associated to a cluster $\mathcal{Y}_i$ (i.e., a state), $i \in |\mathcal{Y}|$
2. Get the set $\mathcal{U}$ of clusters from $(\vec{u}_{(k)})_{k=1}^{K}$ such that each observation $\vec{u}_{(k)}$ is associated to a cluster $\mathcal{U}_j$ (i.e., an input context), $j \in |\mathcal{U}|$
3. Get distribution parameters of B_i from $\{\vec{y}\}$ associated to $\mathcal{Y}_i$
4. Get the state-transition matrix A
   a. the sequence of clusters $(\mathcal{Y}_{(k)})_{k=1}^{K}$ obtained from $(\vec{y}_{(k)})_{k=1}^{K}$ defines the valid state-transitions $A_{ii'}(h)$, $i, i' \in |\mathcal{Y}|$, $h \in \mathcal{U}$
   b. $A_{ii'}(h)$ distribution parameters, $h \in \mathcal{U}$, are computed from $\{\vec{u}\} \in h$.

On the basis of this algorithm, a framework is proposed for investigating drifts in effectiveness of SIS. This framework conceptually takes place in two steps as depicted in Figure 5.10.

❶ The learning algorithm is fed with observations corresponding to the effects expected to be produced by a SIS in different contexts, leading to the learning of a model of its legitimate behaviour.

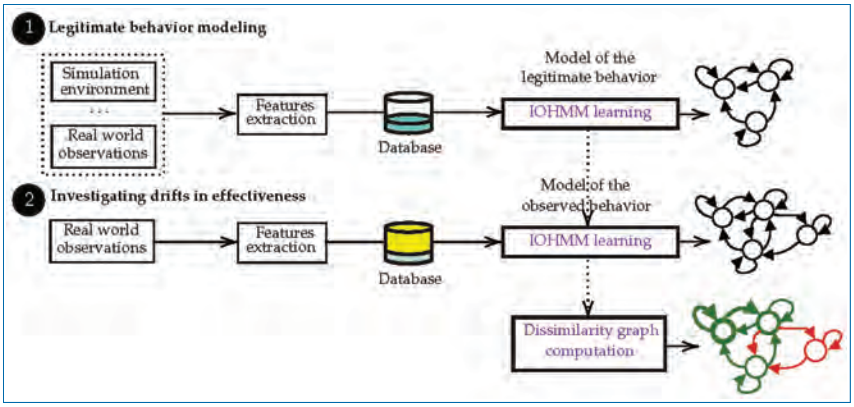

1 Legitimate behavior modeling
Simulation environment
...
Real world observations
Features extraction
Database
Model of the legitimate behavior
IOHMM learning
2 Investigating drifts in effectiveness
Real world observations
Features extraction
Database
Model of the observed behavior
IOHMM learning
Dissimilarity graph computation

Figure 5.10. Steps to implement the proposed approach.

❷ This model is complemented with real-world observations, leading to the learning of a model of the observed behaviour. Then, on the basis of these models, an algorithm is developed to build a directed dissimilarity graph that makes clear the differences between both models, thereby helping designers direct their research and identify the possible causes of drifts in effectiveness.

Let us consider an observation sequence corresponding to the legitimate behaviour, defined by $(\vec{u}^{\star}_{(k)}, \vec{y}^{\star}_{(k)})_{k=1}^{K}$, followed by an observation sequence corresponding to real-world observations, $(\vec{u}_{(k)}, \vec{y}_{(k)})_{k=K+1}^{G}$. The algorithm works as follows (detailed in [29]):

1. Get the set $\mathfrak{Y}$ of clusters from $(\vec{y}_{(k)})_{k=1}^{G}$
2. Get the set $\mathfrak{U}$ of clusters from $(\vec{u}_{(k)})_{k=1}^{G}$
3. Get the set $\mathcal{Y}^{\star} \subset \mathfrak{Y}$ of clusters associated to $(\vec{y}^{\star}_{(k)})_{k=1}^{K}$
4. Get the set $\mathcal{U}^{\star} \subset \mathfrak{U}$ of clusters associated to $(\vec{u}^{\star}_{(k)})_{(k=1)}^{K}$
   ▷ $\mathcal{Y}_i \notin \mathcal{Y}^{\star}, i \in |\mathfrak{Y}|$ corresponds to unexpected states.
   ▷ $\mathcal{U}_i \notin \mathcal{U}^{\star}, i \in |\mathfrak{U}|$ corresponds to unexpected contextual input.
5. The sequence of clusters obtained from $(\vec{y}_{(k)})_{k=1}^{K}$ defines legitimate state-transitions $A^{\star}_{ii'}(h)$, $i, i' \in |\mathcal{Y}^{\star}|$, $h \in \mathcal{U}^{\star}$. Also, the sequence of clusters obtained from $(\vec{y}_{(k)})_{k=K+1}^{G}$ defines observed state-transitions $A_{ii'}(h), i, i' \in |\mathfrak{Y}|, h \in \mathfrak{U}$.
   ▷ State-transitions $A_{ii'}(.)$ defined in the state-transition matrix A but not defined in the state-transition matrix $A^{\star}$ correspond to unexpected state-transitions.
   ▷ State-transitions $A_{ii'}(h)$ defined in the state-transition matrix A and in the state-transition matrix $A^{\star}$ with $h \notin \mathcal{U}^{\star}$ correspond to unexpected state-transitions (i.e., are triggered by unexpected contextual input).
6. Plot the dissimilarity graph where expected states (nodes) and state-transitions (edges) are coloured in green while unexpected ones are coloured in red as depicted in Fig. 5.10.

The dissimilarity graph, without identifying the causes of the drifts in effectiveness, sheds light on the symptoms likely to explain them.

5.4 Smart Home Use-Case and Illustration

The purpose of this section is to illustrate the toolset presented in this chapter and its benefits throughout DevOps life cycles. The illustration is carried out on a real world smart-home scenario, providing the reader with technical details and key results.

5.4.1 Smart Home use Case Description

The experimentation is built upon the widely spread three-layer IoT architecture [17]. The first layer (*Perception Layer*) at the edge of the infrastructure consists of 57 IoT devices providing up to 319 parameters including 249 sensor/actuator whose values/states are updated on a regular basis on a centralized time series database (Synology DS418 Network Attached Storage (NAS) + InfluxDB [23]). IoT devices part of this infrastructure layer are mainly consumer electronics devices (e.g., Netatmo devices, Neato vacuum cleaner, LG TV, etc.), complemented with custom IoT devices based on Arduino Uno/Nano and Raspberry Pi equipped with sensors and actuators targeting specific purposes. The second layer (*Network layer*) enables data transmission and processing, throughout the different IoT wireless communication protocols (Wifi, ZWave [38], etc.). OpenHab [5] and MQTT middlewares provide this layer with the functionalities required for the illustration. Finally, the third layer (*Application layer*) consists in several software components controlling actuators and processing sensor data. These software components (along with middlewares) are encapsulated into docker containers hosted on computational resources (Raspberry Pi) at the edge of the infrastructure.

To illustrate SIS actuation conflict management (Paragraph 5.2) and behavioural assessment/drift analysis (Paragraph 5.3) toolsets, the following devices are used; a smart phone, a TV along with its remote control, a light bulb, an Amazon echo smart speaker and an autonomous smart vacuum cleaner. These devices are all located in the living-room and are implemented as part of a scenario focusing on inhabitants well-being. In this scenario, user-comfort is driven by luminosity and sound physical properties.

A first application set (*UserComfortApps*) is dedicated to give sensor data access to decision making algorithms (*App_Lum* and *App_RC_TV*) that respectively control (1) the luminosity level by acting on the roller shutters and the light bulb and (2) the sound by acting on the TV remote controller and the Amazon echo smart speaker. A second application set (i.e. *CommunicationCenterApps*) is deployed, 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). All applications are implemented through Node-RED flows.

To illustrate the toolsets proposed within the DevOps approach, a scenario with two DevOps cycles has been defined, each cycle covering the development and the operational stages.

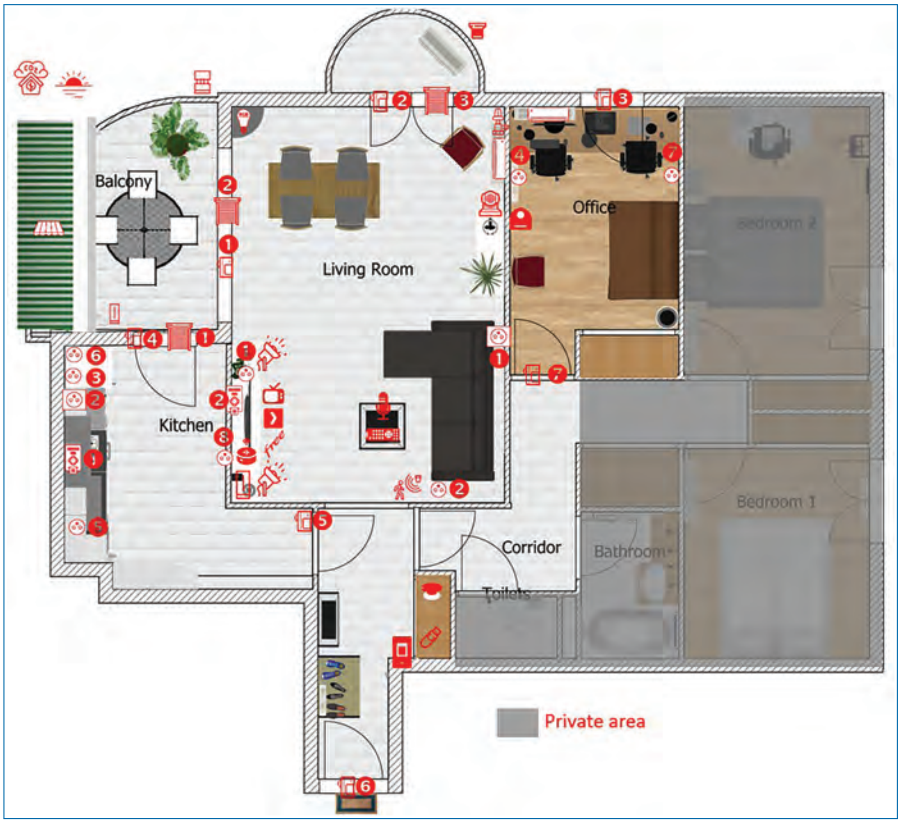


Figure 5.11. Smart Home use case setup.

5.4.2 Software Development (Devs, Cycle 1)

At this stage, the aforementioned software components are designed and developed (Fig. 5.12). Before deploying the whole system, the toolset developed for identifying and resolving actuation conflicts (Paragraph 5.2.2) is applied on the WIMAC model built from deployment, implementation and physical environment models. Designers specify the physical environment model by linking ActionComponents to the PhysicalProperty they act on. At this point, a potential direct actuation conflict is identified on *App_RC_TV* and *App_Phone_TV*; these applications being meant to control the TV sound source (Fig. 5.13).

The management of this direct actuation conflict is straightforward, applications merely send a Boolean value to a shared actuator (ActionComponent). The developer can then select, among the available off-the-shelf ACMs, the one relevant to resolve this generic conflict type (here, a simple OR logic ACM has to be instantiated between both software components (*App_Phone_TV* and *App_RC_TV*) and the actuator they act on (TV)). Once the selected ACM is instantiated into the

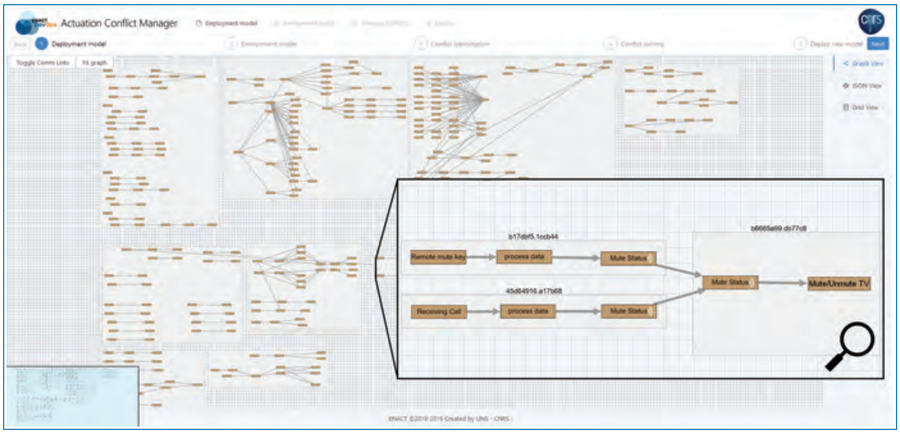


Figure 5.12. Excerpt of *App_Phone_TV* and *App_RC_TV* structural inter-relationships, as described in the WIMAC model, without environment model description.

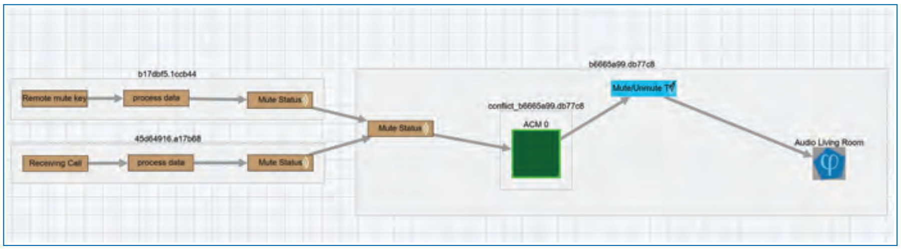


Figure 5.13. Excerpt of *App_Phone_TV* and *App_RC_TV* structural inter-relationships, as described in the WIMAC model with environment model description and the direct ACM instantiated.

WIMAC model, a new deployment model is generated from WIMAC and pushed to GeneSIS which concretizes the deployment.

5.4.3 System Operations (Ops, Cycle 1)

Throughout the execution of the SIS, the effects produced in the physical environment are observed and, thanks to the behavioural drift assessment toolset, the effectiveness of the SIS is measured from a model of the legitimate behaviour it has to comply with (Fig. 5.14).

The model of the legitimate behaviour is built upon a set of sound features computed from a microphone signal. Output observations (expected effects to be produced by the SIS) are then characterized by a Mel-Frequency Cepstral Coefficient[3]

3. https://en.wikipedia.org/wiki/Mel-frequency_cepstrum

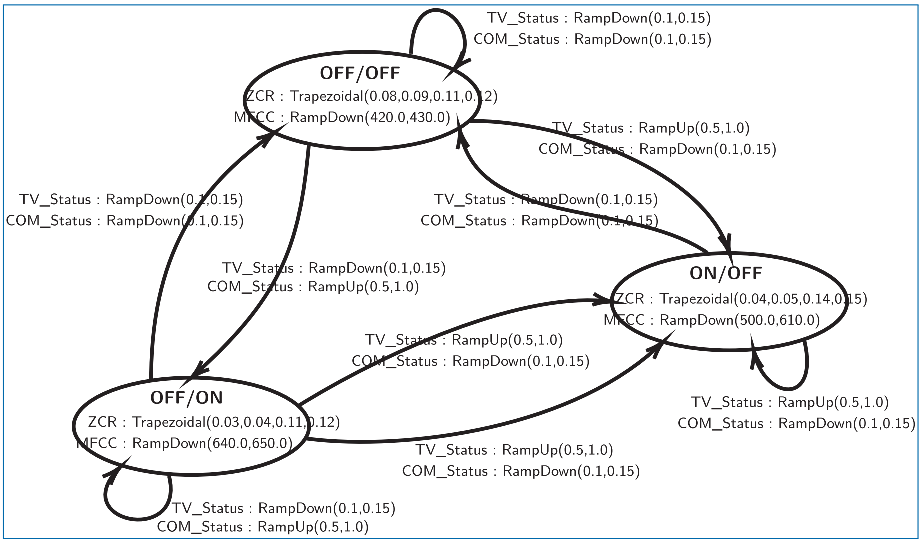

TV_Status : RampDown(0.1,0.15)
COM_Status : RampDown(0.1,0.15)
OFF/OFF
ZCR : Trapezoidal(0.08,0.09,0.11,0.12)
MFCC : RampDown(420.0,430.0)
TV_Status : RampUp(0.5,1.0)
COM_Status : RampDown(0.1,0.15)
TV_Status : RampDown(0.1,0.15)
COM_Status : RampDown(0.1,0.15)
TV_Status : RampDown(0.1,0.15)
COM_Status : RampDown(0.1,0.15)
TV_Status : RampDown(0.1,0.15)
COM_Status : RampUp(0.5,1.0)
ON/OFF
ZCR : Trapezoidal(0.04,0.05,0.14,0.15)
MFCC : RampDown(500.0,610.0)
TV_Status : RampUp(0.5,1.0)
COM_Status : RampDown(0.1,0.15)
OFF/ON
ZCR : Trapezoidal(0.03,0.04,0.11,0.12)
MFCC : RampDown(640.0,650.0)
TV_Status : RampUp(0.5,1.0)
COM_Status : RampDown(0.1,0.15)
TV_Status : RampUp(0.5,1.0)
COM_Status : RampDown(0.1,0.15)
TV_Status : RampDown(0.1,0.15)
COM_Status : RampUp(0.5,1.0)

Figure 5.14. Possibilistic model of the legitimate behaviour. States are defined by the operating status of the TV and the communication status (i.e., OFF/ON means TV:ON, COM:OFF). The configuration ON/ON is not legitimate, the direct ACM instantiated is supposed to mute the TV while a communication is in progress.

(MFCCs) and the Zero Crossing Rate[4] (ZCR) sound features. These sound features lead a good segmentation of the observation space thereby, allow identifying device states on the basis of the sound they emit. Contextual inputs are characterized by the operating status of the TV and the communication status.

As depicted in Fig. 5.15, the ACM instantiated during the design phase fulfils its role and prevents the TV to operate while a communication is in progress. No behavioural drift is therefore reported.

While the SIS operates, free of unexpected effects produced in the physical environment, 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 leading behavioural drifts to occur (Fig. 5.16).

From the designers' point of view, the quantitative effectiveness assessment is not informative on the reasons for the drifts in effectiveness observed. At that point, the behavioural drift analysis tool (Paragraph 5.3.2 and 5.3.3) is leveraged to guide designers and help them correlate the symptoms of the drifts in effectiveness to various events (Fig. 5.17).

On the basis of the dissimilarity graph depicted in Fig. 5.17, designers can infer that drifts in effectiveness are the result of the smart vacuum cleaner unexpectedly

4. https://en.wikipedia.org/wiki/Zero-crossing_rate

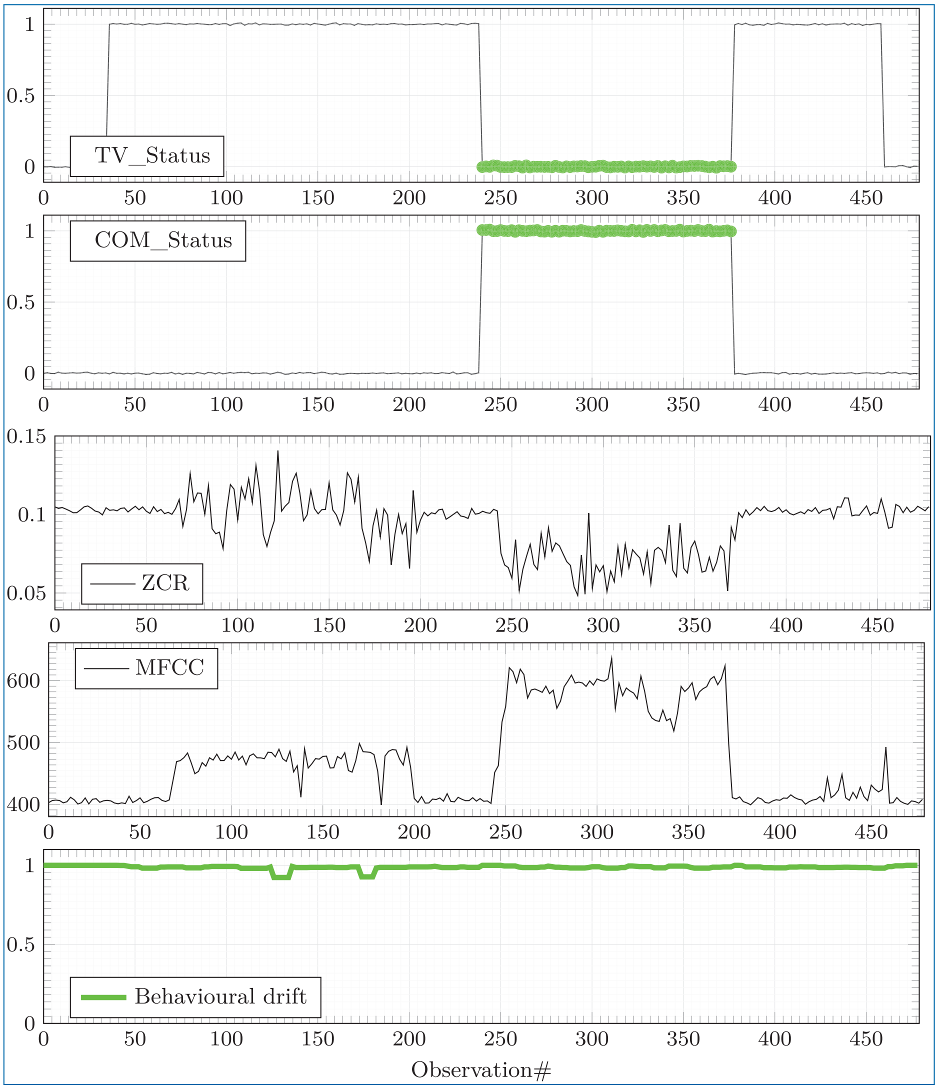


Figure 5.15. Observations from the field corresponding to the legitimate behaviour. The ACM instantiated during the development phase fulfils its role (green part of the graph) and no drift is reported here, the SIS behaves as expected.

(but legitimately) operating in the living-room and producing conflicting noise. This device was unforeseen by designers in the initial model of the legitimate behaviour. Moreover, this device has to be controlled so as to be integrated into the user-well being control strategy and prevent indirect conflicts to occur on the sound physical property.

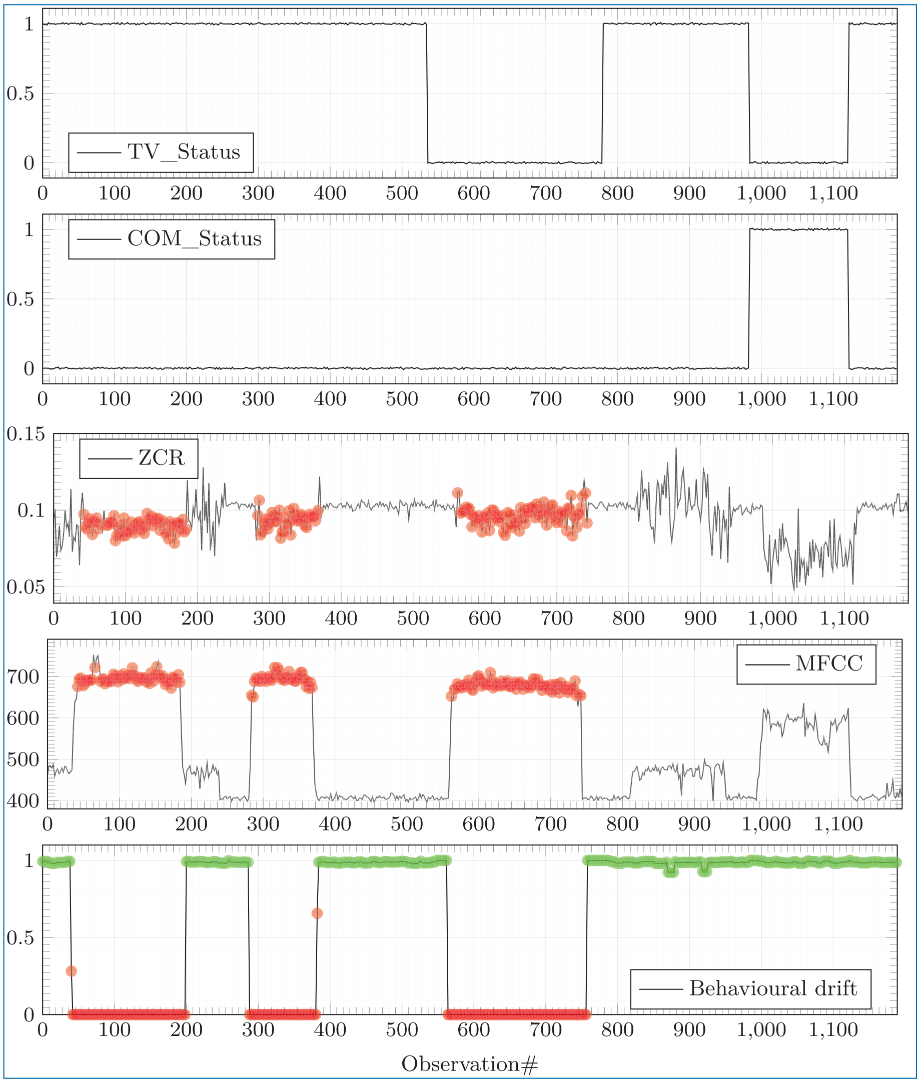


Figure 5.16. Observations from the field coloured in red denote the presence of the smart vacuum cleaner not anticipated/foreseen in the model of the legitimate behaviour (leading behavioural drifts to occur).

5.4.4 Software Development (Devs, Cycle 2)

The behavioural drift reported at run-time suggests that the model of the physical environment is incomplete. Indeed, the indirect conflict on the sound physical property should have been identified during the first phase of development. A new development cycle is therefore necessary to correct the model of the physical environment. The updated model is depicted in Fig. 5.18; an indirect

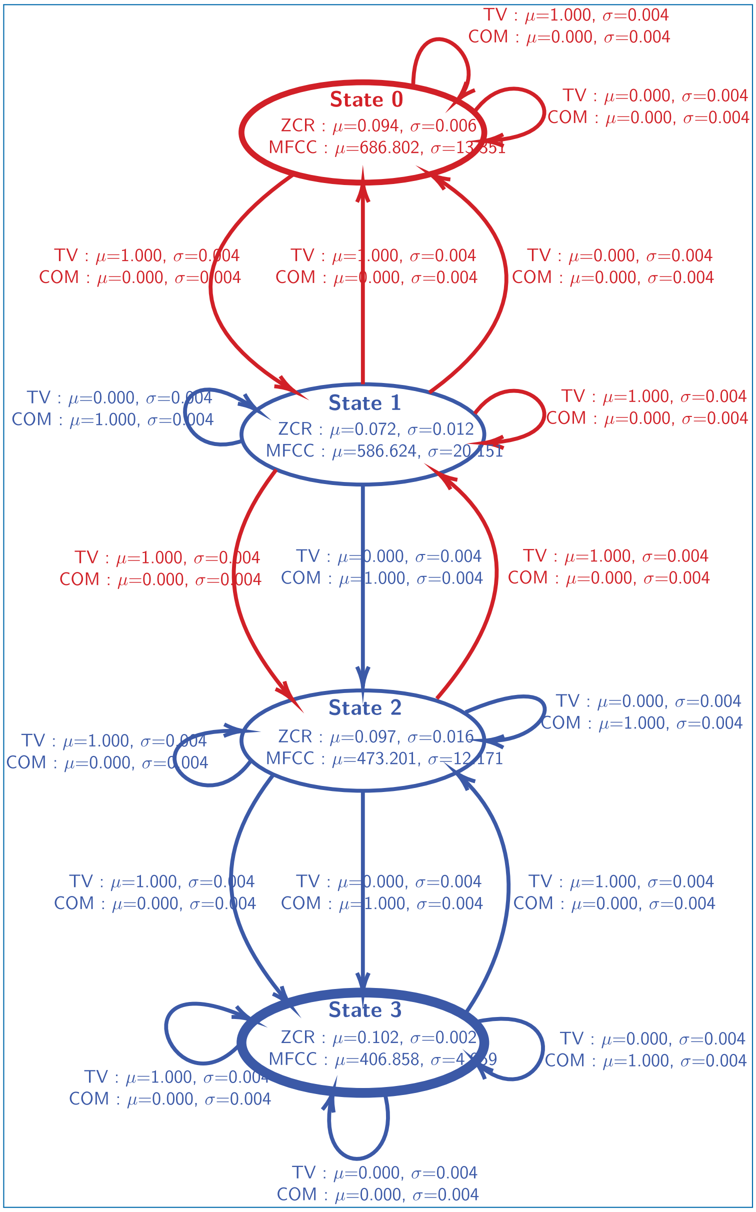


Figure 5.17. Dissimilarity graph. Nodes and edges coloured in blue correspond to the legitimate behaviour, those in red correspond to unexpected behaviour resulting from the appearance of the smart vacuum cleaner in the living-room.

actuation conflict is now detected between the *App_RC_TV*, *App_Phone_TV* and the *App_Vaccum_Cleaner* applications and a dummy ACM has been instantiated.

The management of this indirect actuation conflict implies semantic concerns that cannot be managed through generic off-the-shelf ACMs. A custom ACM

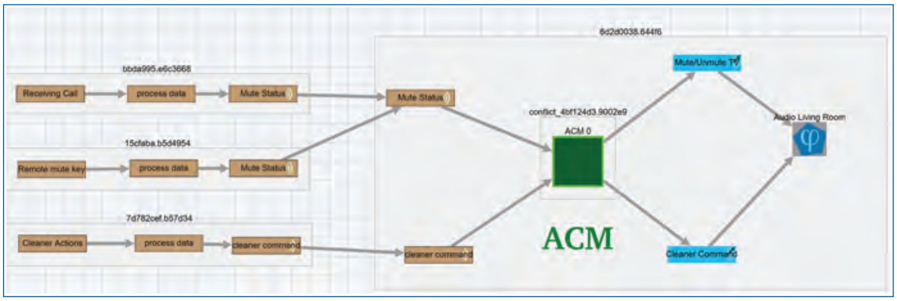


Figure 5.18. Excerpt of indirect conflict detection and management.

has to be developed. This task relies on “ECA” rules that define the ACM logic, further transformed into an FSM. More specifically, designers have to describe a control strategy that prevent the TV and the smart vacuum cleaner robot to produce sound simultaneously. The custom ACM thus receives commands from *App_RC_TV*), *App_Phone_TV* and *App_Vaccum_Cleaner*. On the basis of these input, the ACM control logic defines a strategy resulting in sending commands to the TV and the vacuum cleaner such that it is stopped while a communication is in progress or while inhabitants watch TV. The ACM control strategy defines some prioritization whose semantic has to be defined by designers.

Along with the logical strategy of the custom ACM, some logical and temporal properties to be formally verified have to be specified. To this end, ECA rules are extended into ECA+ rules (more details about this language are given in [7]) to provide the model with the properties to be verified by state-of-the-art MDE verification tools.

Logical properties are verified through NuSMV [8], a state of the art model checker. Model checkers are meant to ensure that logical properties are always verified, regardless of the state sequence being executed. Two main types of logical properties are formally verified: *safety* (i.e., something bad will never occur) and *Liveness* (i.e., something good will eventually happen). In the context of ACM design, safety properties ensure that conflicting states will never occur (e.g., the vacuum cleaner will never operate while a communication is in progress).

Temporal properties are verified through DEVS formalism. FSMs are defined by two functions: (i) the state-transition function computes the new state given the previous state $x_{(k-1)}$ at time $k-1$ and the current input $\vec{u}_{(k)}$ at time k, (ii) the output function computes the outputs $\vec{y}_{(k)}$ that solely depend on the state $x_{(k)}$ at time k. DEVS formalism allows an FSM to be encapsulated into DEVS atomic model coupled with a synchronizer managing asynchronous timings on the inputs of the FSM. The ECA+ language provides a syntax to specify asynchronous timings management strategies for the synchronizer. Temporal properties can then be verified for each strategy within a DEVS simulation environment. It is worth noting that asynchronous timing strategies are meant to reproduce the asynchronous timings that govern the different hardware platforms ACMs are supposed to be deployed to, at the edge of the IoT infrastructure.

Once ACMs temporal properties have been verified in DEVS simulated operational contexts, associated DEVS Atomic models (embedding the synchronizer and the logical behaviour) can be directly used for implementation. Indeed, DEVS Atomic models can be translated into high level programming languages (C, C++, C#, Node.js, etc.), embedding a lightweight execution engine (DEVS kernel). This makes ACMs completely portable on the lightweight hardware platforms available at the edge of the IoT infrastructure.

Here again, the WIMAC model modified with the custom ACM is sent to GeneSIS for the deployment on the platforms, thus completing the second development cycle.

5.4.5 System Operations (Ops, Cycle 2)

Finally, SIS is deployed and the physical effects it produces in the physical environment are observed. The model of the legitimate behaviour is modified to take into account the autonomous vacuum cleaner, as learned during the first cycle. No behavioural drift is reported anymore. However, as the physical environment is complex, there are many reasons why the behaviour of the SIS may drift again and trigger a new development cycle:

- the introduction of a new software component that drives an actuator not correctly defined in the model of the physical environment,
- the introduction of a new device producing physical effects in the environment (as the vacuum cleaner in the use-case),
- unexpected changes in the physical environment in which the SIS operate (e.g., a tree growing in front of a window is likely to have an impact on the luminosity of the room in the long term).

5.5 Conclusion and Future Works

This chapter has introduced two innovative toolsets designed to enrich the DevOps eco-system and meant to address an issue that has been poorly addressed, to date, in the development of trustworthy SIS: the management of applications that can interact with their physical environment through actuators.

The first toolset, called "Actuation Conflict Management" toolset (ACM), is to secure the design of these applications at Devs time. The objective is to identify and resolve, through local structural transformations of the SIS, the actuation conflicts that might arise as a result of the actuation effects produced in the physical environment a priori known and limited to its model. While such precautions at Devs time are necessary, they are not sufficient to guarantee that the SIS will always operate as expected in the physical environment. There is a risk of behavioural drift which is observed and quantified at Ops time, thanks to the second toolset called "Behavioural Drift Assessment & Analysis" toolset (BDA). This toolset makes it possible to assess and analyse the differences between the expected and observed effects of the SIS in real-world environment.

Beyond the interest of these toolsets as part of the DevOps methodology, their complementarity represents a major contribution in the realm of trustworthy SIS. Indeed, it is during consecutive DevOps cycles that the contribution of the couple BDA-ACM can be fully appreciated. The BDA toolset provides DevOps team with information on the observed behavioural drifts, thus motivating new development cycles which then results in changes to the model of the physical environment and/or corrections of the applications carried out in and with the ACM toolset. This complementarity is highlighted in this chapter throughout a smart-home usecase which involves two consecutive DevOps cycles to converge towards a SIS with satisfactory behaviour.

It is within the framework of this complementarity that two possible lines of work are foreseen. The first axis aims to reinforce the added-value of the information obtained from the BDA so as to accelerate the SIS/ACMs re-design cycle. Indeed, while the BDA toolset allows to obtain a model of the effects observed in the field and the symptoms of their dissimilarities with the legitimate ones, it is neither informative on the root-causes underlying these differences nor it provides insights to infer the corrections to be made to the model of the physical environment. The second axis is about leveraging behavioural drift measures as rewards towards self-adaptive ACMs automatically containing/mitigating behavioural drifts at run-time.

References

[1] A. Metzger (Ed.). *Software Continuum: Recommendations for ICT Work Programme 2018–2019*. NESSI White Paper, February, 2016.

[2] Abdullah Al Farooq *et al.* "Iotc 2: A formal method approach for detecting conflicts in large scale IoT systems". In: *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE. 2019, pp. 442–447.

[3] Robert C. Armstrong *et al.* "Survey of existing tools for formal verification". In: *SANDIA REPORT SAND2014-20533* (2014).

[4] Yoshua Bengio and Paolo Frasconi. "An input output hmm architecture". In: *Advances in neural information processing systems*, pp. 427–434, 1995.

[5] James Bruce. *Getting Started with OpenHAB Home Automation on Raspberry Pi*. 2015.

[6] Laurent Capocchi *et al.* "DEVSimPy: A collaborative python software for modeling and simulation of DEVS systems". In: *2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE. 2011, pp. 170–175.

[7] Franck Chauvel *et al. Risk-driven Continuous delivery of Trustworthy Smart IoT Systems*. 2020. URL: https://enact-project.eu/deliverables/D2.3.pdf.

[8] Alessandro Cimatti *et al.* "Nusmv 2: An opensource tool for symbolic model checking". In: *International Conference on Computer Aided Verification*. Springer. 2002, pp. 359–364.

[9] European Commission. *Horizon 2020 Work Programme 2016–2017 – Cross-cutting activities (Focus Areas), Call IoT-03-2017 : R&I on IoT integration and platforms*. Oct. 2015. URL: %7Bhttp://ec.europa.eu/newsroom/dae/document.cfm?%5C&doc%5C_id=11867%7D.

[10] Muhammad Fahim and Alberto Sillitti. "Anomaly detection, analysis and prediction techniques in IoT environment: A systematic literature review". In: *IEEE Access* 7 (2019), pp. 81664–81681.

[11] G. Rocher, J.-Y. Tigli, S. Lavirotte and N. Le Thanh. *Effectiveness assessment of Cyber-Physical Systems*. 2020.

[12] Thibaut Gonnin *et al.* "Actuation Conflict Management Enabler for DevOps in IoT". In: *10th International Conference on the Internet of Things Companion*, IoT '20 Companion. Malmö, Sweden: Association for Computing Machinery, 2020. ISBN: 9781450388207. DOI: 10.1145/3423423.3423474. URL: https://doi.org/10.1145/3423423.3423474.

[13] Edward R. Griffor *et al.* "Framework for cyber-physical systems: Volume 2, working group reports". In: (2017).

[14] Andrew Guillory *et al.* "Learning executable agent behaviors from observation". In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. 2006, pp. 795–797.

[15] Hani Hagras. "Toward human-understandable, explainable AI". In: *Computer* 51.9 (2018), pp. 28–36.

[16] Andrew KS Jardine, Daming Lin, and Dragan Banjevic. "A review on machinery diagnostics and prognostics implementing condition-based maintenance". In: *Mechanical systems and signal processing* 20.7. (2006), pp. 1483–1510.

[17] Mengda Jia *et al.* “Adopting Internet of Things for the development of smart buildings: A review of enabling technologies and applications”. In: *Automation in Construction* 101 (2019) 111–126. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2019.01.023. URL: http://www.sciencedirect.com/science/article/pii/S0926580518307064.

[18] Chris Jones. “Attributed graphs, graph-grammars, and structured modeling”. In: *Annals of Operations Research* 38.1 (1992), pp. 281–324.

[19] Stéphane Lavirotte *et al.* “IoT-based Systems Actuation Conflicts Management Towards DevOps: A Systematic Mapping Study”. In: *IoTBDS*. 2020, pp. 227–234.

[20] Jean Louis Le Moigne. *La modélisation des systèmes complexes*. Bordas Paris, 1990.

[21] Renju Liu *et al.* “RemedIoT: Remedial actions for Internet-of-Things conflicts”. In: *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 2019, pp. 101–110.

[22] Meiyi Ma *et al.* “Detection of runtime conflicts among services in smart cities”. In: *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE. 2016, pp. 1–10.

[23] Mohammad Nasar and Mohammad Abu Kausar. “Suitability of influxdb database for iot applications”. In: *International Journal of Innovative Technology and Exploring Engineering* 8.10 (2019), pp. 1850–1857.

[24] Dang Tu Nguyen *et al.* “IotSan: Fortifying the safety of IoT systems”. In: *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. 2018, pp. 191–203.

[25] Daniella Niyonkuru and Gabriel Wainer. “Towards a DEVS-based operating system”. In: *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. 2015, pp. 101–112.

[26] Lawrence R. Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[27] Gerald Rocher *et al.* “An Actuation Conflicts Management Flow for Smart IoT-based Systems”. In: *2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. Paris, France: IEEE, Dec. 2020, pp. 1–8. ISBN: 978-0-7381-2460-5. DOI: 10.1109/IOTSMS52051.2020.9340196. URL: https://ieeexplore.ieee.org/document/9340196/ (visited on 02/05/2021).

[28] Gérald Rocher *et al.* “A Possibilistic I/O Hidden Semi-Markov Model for Assessing Cyber-Physical Systems Effectiveness”. In: *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. 2018, pp. 1–9.

[29] Gérald Rocher *et al.* “An IOHMM-Based Framework to Investigate Drift in Effectiveness of IoT-Based Systems”. In: *Sensors* 21.2 (2021), p. 527.

[30] Gérald Rocher *et al.* “Overview and Challenges of Ambient Systems, Towards a Constructivist Approach to their Modelling”. In: *arXiv preprint arXiv:2001.09770* (2020).

[31] Claude Rochet. “Public Management as a moral science”. Habilitation à diriger des recherches, Université de droit, d’économie et des sciences – Aix-Marseille III, Dec. 2007.

[32] Alexander Schliep. “A bayesian approach to learning hidden markov model topology with applications to biological sequence analysis”. PhD thesis, Universität zu Köln, 2001.

[33] Trusit Shah *et al.* “Conflict detection in rule based IoT systems”. In: *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE. 2019, pp. 0276–0284.

[34] Antero Taivalsaari and Tommi Mikkonen. “A roadmap to the programmable world: software challenges in the IoT era”. In: *IEEE Software* 34.1 (2017), pp. 72–80.

[35] Arjan J. Van Der Schaft and Johannes Maria Schumacher. *An introduction to hybrid dynamical systems*. Vol. 251. Springer London, 2000.

[36] Yentl Van Tendeloo and Hans Vangheluwe. “An evaluation of DEVS simulation tools”. In: *Simulation* 93.2 (2017), pp. 103–121.

[37] Philippe Weber and Christophe Simon. *Benefits of Bayesian network models*. John Wiley & Sons, 2016.

[38] Muneer Bani Yassein, Wail Mardini, and Ashwaq Khalil. “Smart homes automation using Z-wave protocol”. In: *2016 International Conference on Engineering & MIS (ICEMIS)*. IEEE. 2016, pp. 1–6.

[39] Bernard P Zeigler, Alexandre Muzy, and Ernesto Kofman. *Theory of modeling and simulation: discrete event & iterative system computational foundations*. Academic press, 2018.

[40] Xiangrui Zeng and Junmin Wang. “A stochastic driver pedal behavior model incorporating road information”. In: *IEEE Transactions on Human-Machine Systems* 47.5 (2017), pp. 614–624.

[41] Tao Zheng and Gabriel A Wainer. “Implementing finite state machines using the CD++ toolkit”. In: *Proceedings of the SCS Summer Computer Simulation Conference, 2003. atomic model, 1 CD++, 1 coupled model, 1 DEVS, 1 DEVS Graph, 1 discrete-event modeling*. Citeseer, 2003.

DOI: 10.1561/9781680838251.ch6

Chapter 6

Online Reinforcement Learning for Self-Adaptive Smart IoT Systems

By Alexander Palm, Felix Feit and Andreas Metzger

6.1 Introduction

In this chapter we explain how Reinforcement Learning (RL) techniques can be leveraged to improve the way self-adaptive smart IoT systems (SIS) adapt at run-time.

The concept of self-adaptation facilitates developing software systems that are capable of maintaining their quality requirements even if the systems' environment changes dynamically [3, 18]. Self-adaptation thereby helps developing systems that can operate in a resilient way at run-time. To this end, a self-adaptive software system (such as a self-adaptive SIS) can modify its own structure, parameters and behavior at run-time based on its perception of the environment, of itself and of the fulfilment of its requirements. An example is a self-tuning thermostat for a Heating, Ventilation and Air Conditioning (HVAC) system. Based on its perception of the outdoor and indoor temperature it can control the strength of its heating and cooling devices in order to proactively reach a set point temperature to maximize user comfort. On the other hand it can learn to reduce energy by reducing heating and/or cooling strength when no user is present in the room.

To develop a self-adaptive SIS, software engineers have to develop self-adaptation logic that encodes when and how the system should adapt itself. Software engineers, for instance, may specify event-condition-action rules that determine which adaptation action is executed in response to a given environment change. Developing self-adaptation logic requires an intricate understanding of the software system and its environment, and how adaptations impact on system quality [6, 7]. Among other concerns, it requires anticipating the potential environment changes the system may encounter at run-time to determine how the system should adapt itself in response to these environment changes.

However, anticipating all potential environment changes at design time is in most cases infeasible due to design time uncertainty [7, 17]. In addition, while the principal effects of an adaptation on the system may be known, accurately anticipating the effect of a concrete adaptation is difficult; e.g., due to simplifying assumptions made during design time [7, 10]. One emerging way to address design time uncertainty is to employ online RL [1, 2, 4, 8, 12, 14, 22–24].

Online RL can learn the effectiveness of adaptation actions through interactions with the system's environment. This means that instead of software system engineers having to manually develop the self-adaptation logic, the system automatically learns the self-adaptation logic via machine learning at run-time. The software system engineer expresses the learning problem in a declarative fashion, in terms of the learning goals the system should achieve. Online RL thereby automates the manual engineering task of developing the self-adaptation logic.

Therefore, the remainder of this chapter is structured as follows: Section 6.2 motivates the application of RL in the realm of SIS by briefly introducing the topics of self-adaptive software systems (SASS) and RL. Section 6.3 combines the aforementioned topics and introduces the concept of policy-based RL and explains how it helps to address large continuous state spaces which is a main shortcoming of state-of-the-art approaches leveraging RL at run-time. Section 6.4 shows our experimental results after using our policy-based RL approach for the realization of a self-tuning thermostat in the smart building domain. Section 6.5 exemplifies how the reward function of a RL problem can be decomposed into several reward streams with different semantics to make decisions of an RL agent explainable. Finally, Section 6.7 concludes the chapter.

6.2 Fundamentals

In this section the fundamentals of self-adaptive software systems and Reinforcement learning are briefly introduced.

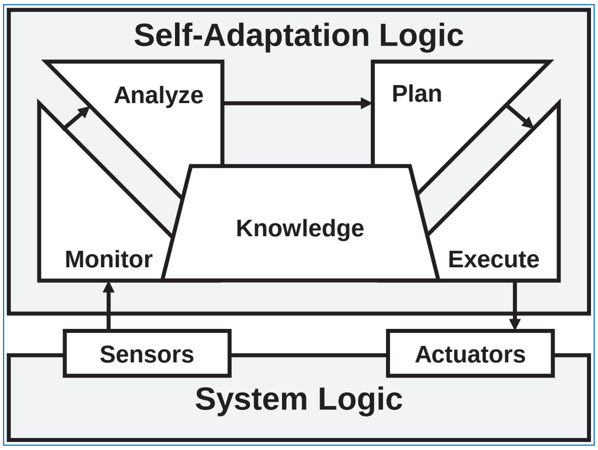


Figure 6.1. MAPE-K reference model for self-adaptive systems (based on [11]).

6.2.1 Self-adaptive Software Systems

A well-known reference model for self-adaptive systems is the MAPE-K model [11], which is depicted in Figure 6.1. Following this reference model, a self-adaptive software system can be logically structured into two main elements: the system logic (aka. the managed element) and the self-adaptation logic (the autonomic manager).

As shown in Figure 6.1, the self-adaptation logic can be further structured into four main conceptual activities that leverage a common knowledge base [9]. The knowledge base includes information about the managed system (e.g., encoded in the form of models at run-time), its environment, and its adaptation goals and adaptation policies (e.g., expressed as rules). The four activities are concerned with monitoring the system logic and the system's environment via sensors, analysing the monitoring data to determine the need for an adaptation, planning adaptation actions, and executing these adaptation actions via actuators, thereby modifying the system logic at run-time.

6.2.2 Reinforcement Learning

RL aims to learn suitable actions via an agent's interactions with its environment [20] as depicted in Figure 6.2. At a given time step t, the agent selects an action a (from its adaptation space) to be executed in environment state s. As a result, the environment transitions to s' at time step $t + 1$ and the agent receives a reward r for executing the action. The reward r together with the information about the next state s' are used to update the knowledge of the agent. The goal of RL is to optimize cumulative rewards. One fundamental problem in RL is the trade-off that must be made between *exploitation* (using current knowledge) and *exploitation*

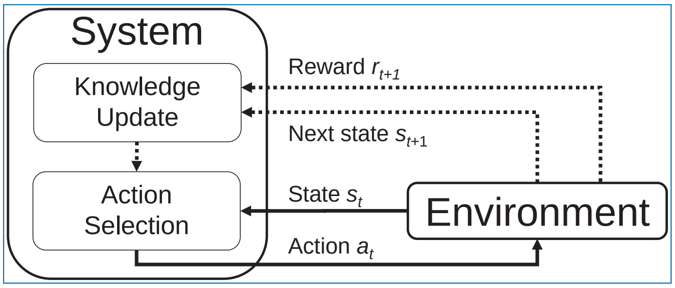


Figure 6.2. Schematic illustration of agent-environment interaction in RL (based on [20]).

(gathering new knowledge). For a self-adaptive service, “agent” refers to the self-adaptation logic of the service and “action” refers to an adaptation action [16].

6.3 OLE: Policy-based Online Reinforcement Learning

This sections introduces our online RL approach. In Section 6.3.1, we provide an overview, the conceptual ideas behind the approach how it differs from the state of the art. In Section 6.3.2, we explain how we prototypically realized the approach.

6.3.1 Overview of Our Approach

The innovative concept underlying the ENACT Online Learning Enabler (OLE) is that we use a fundamentally different type of reinforcement learning than what has been used in the state of the art. While the state of the art used value-based RL, we use policy-based RL. The main idea behind policy-based RL is to directly use and optimize a parametrized stochastic *action selection policy* [15, 21]. The action selection policy maps states to a probability distribution over the action space (i.e., set of possible actions). This means that actions are selected by sampling from this probability distribution. A *learning cycle* consists of a predefined number of *n* time steps. At the end of each learning cycle, the trajectory (comprising the selected *n* actions, states and rewards) are used for a policy update. During a policy update, the policy parameters are perturbed based on the rewards received, such that the resulting probability distribution is shifted towards a direction which increases the likelihood of selecting actions which led to a higher cumulative reward.

Figure 6.3 depicts the conceptual architecture of our approach, showing how the elements of policy-based RL are integrated into the MAPE-K loop. The dark-gray area indicates where the *action selection* of RL takes the place of the *analyze* and *plan* activities of MAPE-K. The learned *stochastic policy* takes the role of the self-adaptive system’s *knowledge* base.

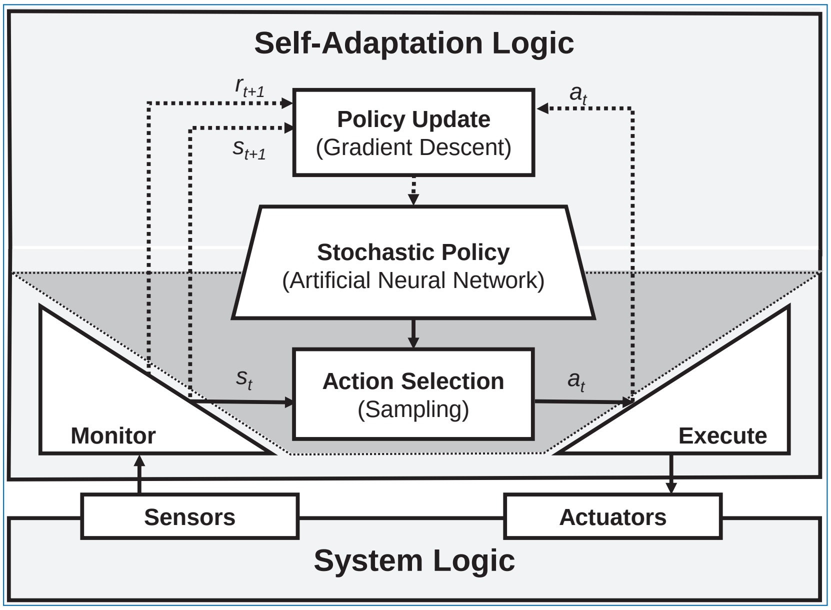


Figure 6.3. Conceptual architecture of policy-based approach.

At run-time the policy is used by the self-adaptation logic to select (via sampling) an adaptation action a_t based on the current state s_t determined by the *monitoring* activity. Action selection determines whether there is a need for an adaptation (given the current state) and plans (i.e., selects) the respective adaptation action to *execute*. In our approach, state s_t may be determined using observations of both the system's environment and the system logic itself. This differs from the basic RL model, where only observations from the system's environment are considered to determine the state s_t. A *policy update* utilizes the trajectory of actions a_t, states s_{t+1}, and rewards r_{t+1} to update the policy. In our approach, policy updates are performed via so-called policy gradient methods [20, 21], because the policy is represented as an artificial neural network. Policy gradient methods update the policy according to the gradient of a given objective function, such as the average reward per learning cycle to give a simple example. In our architecture, rewards are computed by the monitoring activity, as this activity has access to all sensor information collected from the system and its environment.

As mentioned above, the learning problem is stated in a declarative fashion. Typically, it can be formalized as a Markov decision process $MDP = (S, A, T, R)$, with

- S being the state space composed of a set of environment and system states $s \in S$ observable by monitoring via the system logic's sensors (e.g., system workload and performance of the system),

- A being the action space with a set of possible adaptation actions $a \in A$, i.e., possible ways the system may be adapted using the system logic's actuators (e.g., turning off or on different system features),
- $T : S \times A \times S \rightarrow [0, 1]$ being the transition probability among states with $T(s_t, a_t, s_{t+1}) = \Pr(s_{t+1}|s_t, a_t)$, which gives the probability that an adaptation action a_t in state s_t will lead to a state s_{t+1}, and
- $R : S \rightarrow \mathrm{IR}$, being a reward function which specifies the numerical reward the system receives in state s_t. The reward function expresses the learning goal to achieve, which in our case expresses maintaining the quality requirements of the system (e.g., performance should not fall below a given threshold).

Policy-based reinforcement learning finds a solution to the MDP in the form of a parametrized stochastic policy $\pi_\theta : S \times A \rightarrow [0, 1]$, giving the probability of taking adaptation action a in state s, i.e., $\pi_\theta(s, a) = \Pr(a|s)$. The policy's parameters (weights of the artificial neural network) are given as a vector $\theta \in \mathrm{IR}^d$.

Regarding design time uncertainty, we assume that we know A, S, and R, but do not know T. More precisely, even if we do not know the exact states and thus state space S, we know the state variables. As an example, even if we do not know exact workloads of a web application (and maybe not even the maximum workload), we can express a state variable workload $w \in \mathrm{IN}^+$. We assume that we do not know T due to design time uncertainty about how adaptation impacts on system quality. As an example, we may not have an exact understanding of how different configurations of the system perform under different workloads.

6.3.2 Prototypical Realization

To select a concrete policy-based RL algorithm for the implementation of our approach, we took into account two main considerations. First, as we assume we do not know the transition function T, we need to use a model-free variant of policy-based RL. Second, to facilitate online learning, we need an algorithm that continuously updates the policy without waiting for a final outcome, i.e., without waiting for reaching a terminal state. Actor-critic algorithms are a model-free variant of policy-based RL algorithms that use bootstrapping (i.e., knowledge is updated continuously without waiting for a final outcome). We use proximal policy optimization (PPO [19]) as a state-of-the-art actor-critic algorithm. PPO is rather robust for what concerns hyper-parameter settings. Thereby, we avoid extensive hyper-parameter tuning compared to other actor-critic algorithms. In addition, PPO avoids too large policy updates by using a so called clipping function. A too large policy update may mean that RL misses the global optimum and remains stuck in a local optimum. To represent the actor and critic models of PPO, we used multi-layer perceptrons with two hidden layers of 64 neurons each (neurons in the input and output layers depended on the respective number of action and state variables).

6.4 Validation in the Smart Building Domain

To show the applicability of our policy-based RL approach on SIS we performed a series of experimental validations in the smart building domain (cf. Section 10.5). Therefore, the experimental setup is summarized in Section 6.4.1 before the underlying RL problem is formalized as a MDP in Section 6.4.2. Finally, Section 6.4.3 shows the results of our experiments.

6.4.1 Experimental Setup

In the according use case scenario we show that it is possible to learn a control strategy for a simulated HVAC system by means of policy-based RL. The simulated HVAC system is based on the ground floor of the KUBIK building (cf. Section 10.5) and comprises six multi-sensors providing information about user presence and temperature, as well as 5 fan coils whose capacity is accumulated to treat them as one single fan coil. An excerpt of the KUBIK specification showing the room used for the simulation is depicted in Figure 6.4.

The learned control strategy thereby should control the HVAC system in such a way that thermal comfort is achieved whenever the controlled room is occupied and energy consumption is minimized otherwise. The run-time of each experiment corresponds to one year of simulation, while one time-step of an experiment corresponds to one minute (i.e. total time-steps of one experiment: 524000). After several iterations of code improvements of the simulation we were able to simulate

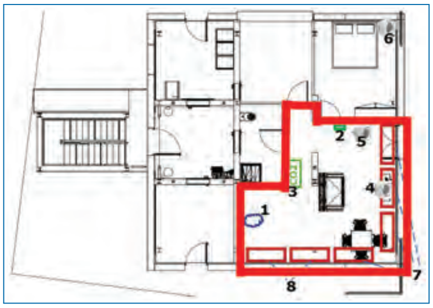


Figure 6.4. Excerpt of the KUBIK specification showing the room used for the simulation.

between 150 and 1000 time-steps per seconds depending on the hyperparameters (especially the size of the neural net) of the underlying algorithm.

As a baseline we implemented a simple on/off-controller, that heats or cools the room whenever the indoor temperature is not close enough to the user set-point and a user is present. If no user is present the thermostat controller remains inactive.

6.4.2 Problem Formalization as MDP

For the evaluation, in a first step we formulated the problem of learning an HVAC control strategy as a RL problem. The underlying Markov Decision Process (MDP) is thereby specified as follows:

State-space: The main state variables are stemming from the from simulation variables (e.g. indoor temperature). Furthermore, we created some crafted features resulting in variables relying on main state variables (e.g. deviation from setpoint). The variable predicted occupancy returns the probability that the room gets occupied within the next 30 min. This variable is computed based on the underlying occupancy pattern, as we assume that such variables might exist in modern HVAC systems (cf. [5]).

Action-space: As the main task of the control strategy is to properly control the HVAC device, the action-space comprises 7 discrete actions, where only one action could be selected at a time. The actions correspond to the different modes of the fan coil for heating and cooling, as well as a turning he device off. The different modes relate to different fan speeds resulting in different air flow rates and different temperatures concerning the integrated fluids for heating or cooling respectively.

Transition-dynamics: The transitions between the different states (depending on the selected action) are computed by the simulation according to the underlying thermal equations. As we employ model-free RL algorithms, the control strategy does not have access to the environmental model resulting from the simulation. It is learning through pure interaction with raw experience. The simulation could be treated as a real environment, with the main difference that with a real environment the run-time of an experiment would be much longer.

Reward: The main part of the MDP driving the algorithm in a direction to learn the right control strategy is the reward function. We defined the reward in such a way, that the algorithm gets a positive feedback whenever its control strategy keeps the indoor temperature close (i.e. within bounds of 1°C) to the user setpoint when a user is present and penalized otherwise (i.e. negative feedback corresponding to the deviation from the setpoint). As the control strategy should minimize energy consumption, the algorithm gets penalized according to the strength of the current action, whenever it performs an action other than turning the HVAC system off, if no user is present in the room (based on the simulated occupancy).

6.4.3 Results

We evaluate the results from two different perspectives: The domain perspective and the Reinforcement Learning perspective. For the domain perspective, we plotted the outdoor temperature curve and the indoor temperature curve showing the moving average of both variables. For the moving average we averaged the last 128 values to reduce the noise inside the diagram and improve the interpretability of the results. Furthermore, we plotted the average indoor temperature per occupancy phase. This gives us the option to see whether the learned control strategy is able to avoid heating or cooling actions in occupancy phases where no user is present and to keep the indoor temperature close to the user set-point whenever a user is present. Apart from this domain-related metric, we used the moving average reward of the last 10000 time-steps as a metric to evaluate the learning process from a RL-perspective. It is important to note that the values of the average reward are scaled to fit into the temperature diagram. The scaling factor is neglectable, as it is only the evolution of the reward curve that is important in this case. The exact evolution of the cumulative reward is shown in different plots to address solely the Reinforcement Learning perspective, when we compare the results of the RL approach to the baseline approach.

Figure 6.5 shows how the indoor temperature evolves according to the control strategy resulting from the baseline thermostat. As can be seen during the first

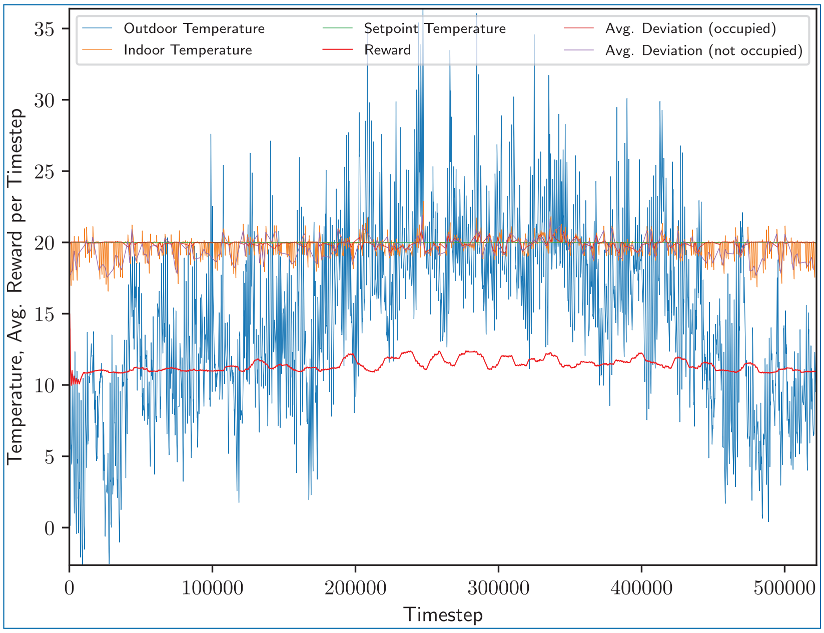


Figure 6.5. Temperature evolution of baseline thermostat.

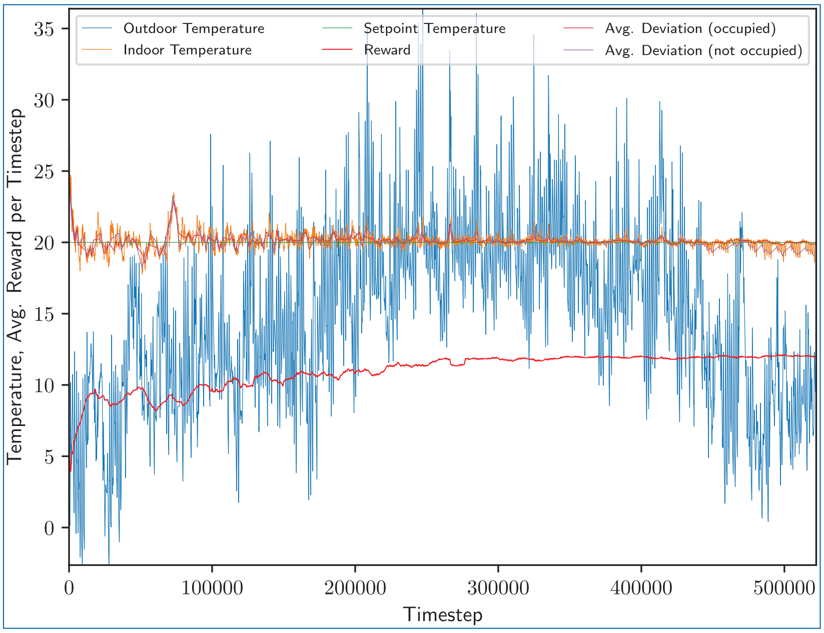


Figure 6.6. Temperature evolution of RL approach.

150.000 time steps, the indoor temperature drops if no user is present resulting from the heating device being turned off while the outdoor temperature is below the set point temperature. As the outdoor temperature becomes warmer (from time step 150.000 onwards) the indoor temperature is higher than the set point temperature (if not user is present) as the cooling device is turned off accordingly. In phases where a user is present the indoor temperature is kept around the set point temperature. However, from a RL perspective this leads to a non-optimal reward, because the thermostat controller is purely reactive and for every time step the indoor temperature is too far from the set point temperature this leads to a non-optimal reward.

In contrast, Fig. 6.6 shows how the indoor temperature evolves to a thermostat controller based on a policy-based RL approach. After a learning phase (until time step 260.000) the RL approach is able to keep the indoor temperature around the set point temperature if a user is present and reduce energy consumption by turning off the heating or cooling device otherwise. Especially during the end of the experiment, after learning has been converged and the approach can reuse its knowledge about low outdoor temperatures (from time step 450.000 onward), it can be seen that the same spikes can be observed as in 6.5. However, the drops in the indoor temperature are not as big as with the baseline approach and the reward is slightly higher.

Figure 6.7 shows the results from the RL perspective visualizing the cumulative reward evolution. The baseline approach outperforms the RL approach in terms of

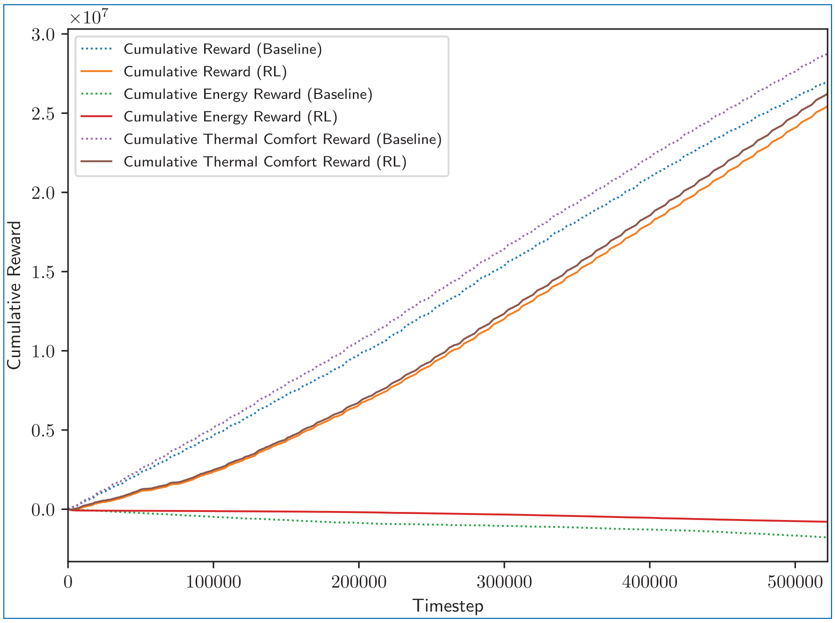


Figure 6.7. Cumulative reward (baseline vs. RL approach).

cumulative reward, resulting from the poor initial performance during the learning phase. This is due to the fact that the RL approach has no initial knowledge about a goal-directed behavior. However, after learning has converged (about time step 260.000) the increase in reward becomes more steep than that of the baseline approach, which can be seen from time step 400.000 onward.

After having shown that the RL approach is able to outperform the baseline approach after its learning process has converged, we did further investigate how it may perform if its knowledge has been initialized a priori. To do this we did set up a slightly modified version of the HVAC simulation, with the thermal dynamics being simplified (and not following complex thermal equations). To perform some kind of pretraining we let the RL approach learn with this simplified environment for the same amount of time steps as in the online experiment.

As it can be seen in Fig. 6.8 the pretrained version of the RL approach is able to adapt its control strategy to the real thermal dynamics pretty fast and after around 50.000 time steps the reward curve converges. This results from a goal-directed behavior with the pretrained RL approach being able to keep the indoor temperature around the set point temperature whenever a person is present and reduce energy consumption otherwise. The reason for the slightly better reward compared to the baseline approach can be seen in Fig. 6.9. The RL approach learns to avoid the indoor temperature moving too far from the set point temperature, to be able to reach the set point temperature within fewer time steps than the baseline approach.

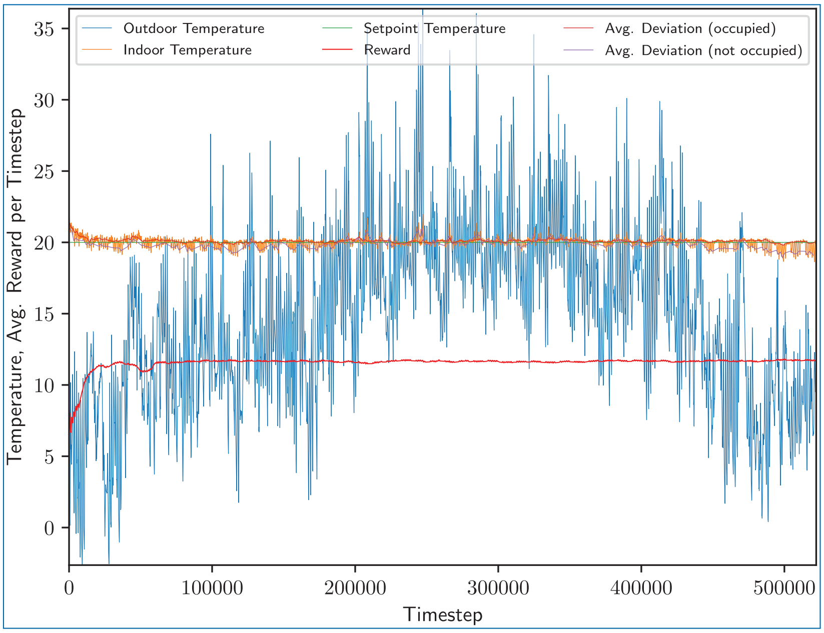


Figure 6.8. Temperature evolution of pretrained RL approach.

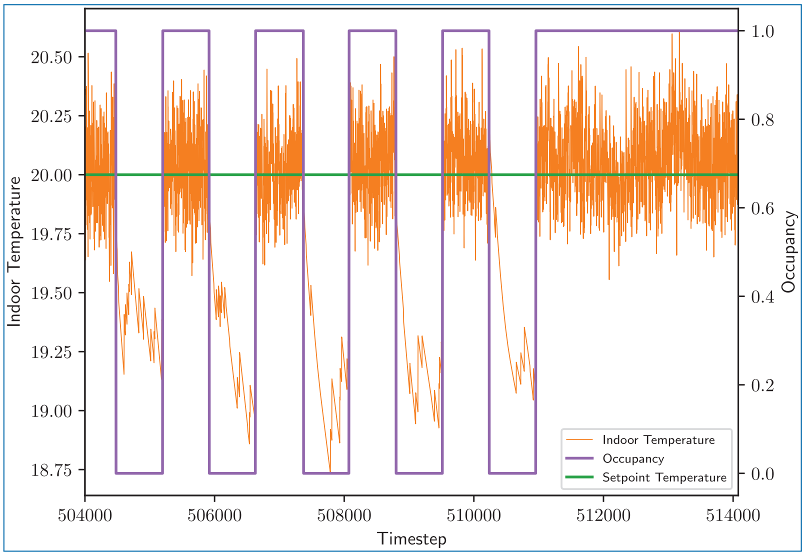


Figure 6.9. Temperature evolution of baseline thermostat.

This is done by proactively heating or cooling resulting in small penalty for consuming energy while the room is not occupied. However, this penalty is rather small to the penalty that would be received during a time step (with the room being occupied) where the indoor temperature is not around the desired set point temperature.

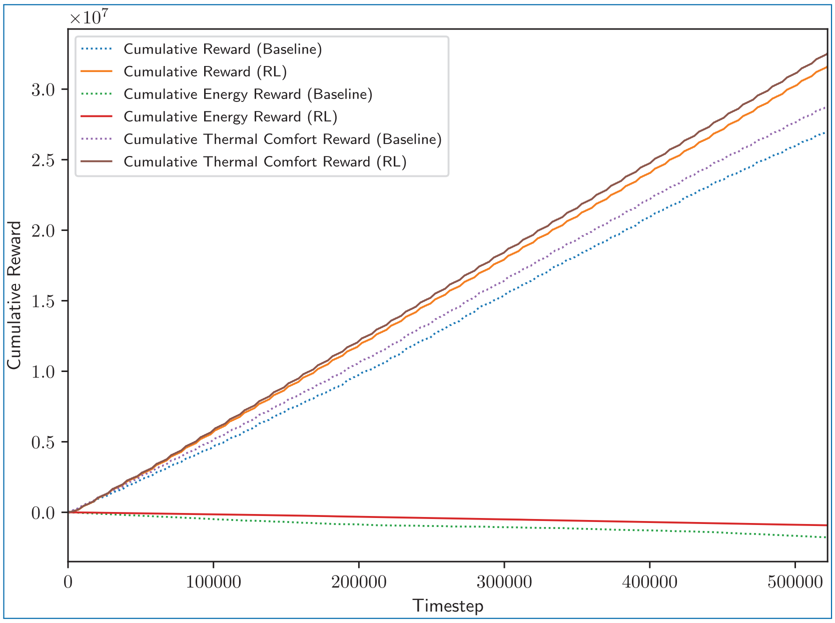


Figure 6.10. Cumulative reward (baseline vs. pretrained RL approach).

Finally, when having a look at the evolution of the cumulative reward as depicted in Fig. 6.10, the pretrained RL approach clearly outperforms the baseline approach.

6.5 Explaining Adaption Decisions via Reward Decomposition

Reward Decomposition is a method in which several value-based RL agents are trained in parallel on different aspects of an environment. At each time step the knowledge of the agents is aggregated to provide a global decision. To apply this method, it is necessary that the reward function of the examined environment can be decomposed into independent subfunctions. As a result, instead of returning a scalar value at each time step, the reward function returns a vector where each component reflects the reward of one subfunction or zero if there was no reward or punishment at the associated time step.

For each component of the reward vector an independent subagent is trained, which receives the global state as observation and as reward only one component of the reward vector. In order to derive the action of the overall agent, at each time step the action values of all subagents are summed up element-wise and on the basis of these aggregated action values an action is selected (e.g. by using epsilon greedy action selection).

This technique was applied to a simplified version of the HVAC environment (see Chapter 6.4) where the cooling capability was removed. In this environment the reward was split into the two subfunctions “thermal comfort” and “energy cost”. The first of these functions always gives a negative value, i.e. a penalty if the person is present but the current temperature is not within a certain tolerance around the desired temperature. On the other hand, the second component “energy costs” contains a constant negative value if action “heating” was chosen in the last step. One value-based RL agent (e.g. DQN) is then trained for each of these two components of the reward vector. At each time step the action values of both agents are summed up for both actions “heating” and “not heating”. The greater sum then marks the greedy action of the overall agent.

To generate explanations for actions, the action-values of the subagents can be put in relation to each other. For this purpose, the difference between the action-value of a particular action and the action-values of all alternative actions is calculated for each subagent. This is repeated for all actions and yields the relative importance per action per subagent. The higher the relative importance of an action, the more influence the subagent has on the selection of this action. By observing the behaviour and relative importance the reasoning of an agent can be deduced.

To further increase the explainability of the approach, the reward function can be broken down into situations instead of subfunctions. Each situation represents a set of special states of the environment. In addition, each situation receives a non-zero reward or punishment value that is always given when the agent is in that situation and zero otherwise. For example, the HVAC environment can be deconstructed into the following four situations:

- occupied and within the tolerance (reward: $+1$)
- occupied and out of tolerance (reward: -5)
- unoccupied and within the tolerance (reward: -1)
- unoccupied and out of tolerance (reward: $+0.1$)

For each of these situations, as before, a separate agent is trained and the relative importance is calculated. If the importance is then set in relation to the sum of the relative importance of all actions, the relative importance of an action in relation to a specific situation can be calculated. These values can then be summed up to obtain the relative importance of an action across all situations. Using these two metrics, the following natural language string can be generated for each time step:

“With my current knowledge I am 97% sure that action ‘not heating’ is better. Arguments in favour of action ‘not heating’ are the prevention of situation ‘occupied and out of tolerance’ (84%), the occurrence of situation ‘occupied and within the tolerance’ (9%), and the prevention of situation ‘unoccupied and within the tolerance' (4%). An argument in favour of action 'heating' is the occurrence of situation 'unoccupied and out of tolerance' (2%)."

The first sentence of this explanatory string contains the relative importance of an action across all situations (97%) and the following sentences describe the relative importance of an action in relation to a specific situation (84%, 9%, 4%, and 2%).

6.6 Synergies with Behavioural Drift Analysis

The main goal of our Online Reinforcement Learning approach is to learn an optimal control strategy for a MDP. Despite of evaluating the learned control strategy from a RL perspective, it needs to be evaluated from a domain perspective as well. The latter can also be used to guide the engineering of the reward function. As the behavioral drift analysis is based on a model capturing the desired control strategy in an abstract way, the computed signal might be used for the evaluation of the actual control strategy. Unexpected changes in the behavioral drift signal can then be interpreted as an indicator for context changes that make it impossible for the learning system to behave according to the obtained model. To showcase this synergy we derived a model for BDA based on the desired behavior described in Section 6.4.2 (cf. Figure 6.11).

The behavioural model depicted in Figure 6.11 represents the expected indoor temperature which depends on whether or not a person is present in the room and this, independently of the underlying temperature management system. It relies

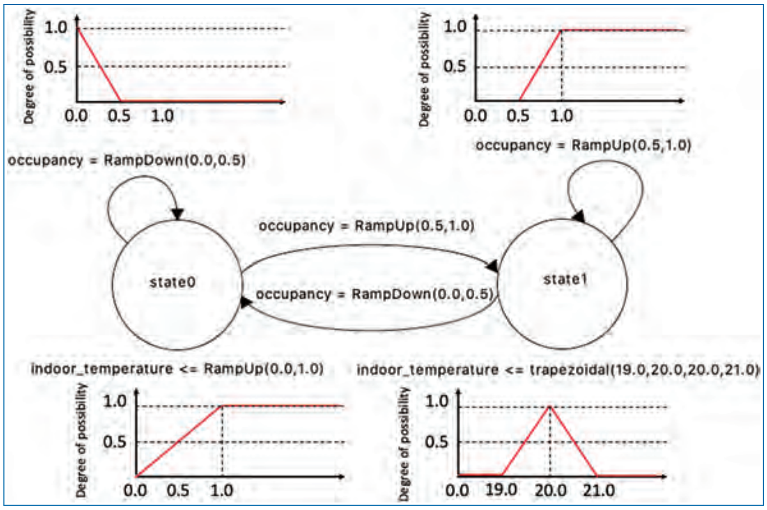

Degree of possibility
1.0
0.5
0.0 0.5 1.0
occupancy = RampDown(0.0,0.5)
occupancy = RampUp(0.5,1.0)
occupancy = RampUp(0.5,1.0)
occupancy = RampDown(0.0,0.5)
state0
state1
indoor_temperature <= RampUp(0.0,1.0)
indoor_temperature <= trapezoidal(19.0,20.0,20.0,21.0)
0.0 19.0 20.0 21.0

Figure 6.11. Possibilistic Input/Output Hidden Markov Model describing the expected indoor temperature depending on whether a person is present or no in a room.

on the possibility theory where distributions are defined as membership functions. The model defines the expected behaviour as follows: when a person is present in the room, the temperature must be equal to 20°C (state 1). When nobody is in the room, the temperature is expected to be greater than 1°C (state 0). In addition to fully accepted temperature values (where degree of possibility is equal to one), the model defines some tolerances. For instance, for the state 1, temperatures below 19.0°C and above 21°C are totally rejected (degree of possibility = 0) while temperatures in between 19.0°C and 21.0°C different from 20°C, while not being perfect, are not totally rejected (0< degree of possibility <1). In conjunction with temperature and occupancy sensor values, this possibilistic Input/Output Hidden Markov Model (IOHMM) is used to compute the behavioural drift as the likelihood (possibility measure) of the observation sequences to have been generated by the model, i.e. the likelihood that the temperature is managed in such a way that it remains within the accepted boundaries defined by the model.

6.7 Conclusion and Outlook

We motivated the application of Reinforcement Learning as a means to enable a software system to adapt itself to changing context situations in the realm of SIS. Furthermore we introduced a concrete realization of an Online Learning approach which overcomes the main shortcomings of state-of-the-art approaches (e.g. ability to handle continuous parameters as actions and avoid manual fine-tuning of exploration). Our policy-based Reinforcement Learning approach for a self-adaptation logic has been validated in the smart building domain by applying it to an HVAC control problem. The experiment results have shown that our approach is able to outperform static thermostat implementations by dynamically learning to control the heating and cooling devices of a smart building. This has been achieved by finding a trade-off between the maximization of user comfort and minimization of energy consumption. Additionally, we introduced our conceptual work on the process of decomposing a reward function of a RL problem into several reward streams with different semantics to make decisions of an RL agent explainable and proposed how our Online Learning approach can be enriched by the concept of Behavioral Drift Analysis.

As future work, we envision extending our approach for online reinforcement learning for self-adaptive Smart IoT Systems along the following two main dimensions:

Better Pre-training As we demonstrated above, pre-training the reinforcement learning enabler may deliver better performance during operations. On the one hand, the initial performance (directly after deployment to run-time) can be increased. On the other hand, the overall speed of learning and learning performance can be increased, in particular in real-world situations where rewards are sparse. Yet, such offline pre-training again faces the uncertainty issue when formulating the source learning task to be learned in the offline setting. It is not possible due to design time uncertainty that this source learning task faithfully captures the actual online setting. To capture the problem of uncertainty, existing solutions thus make certain assumptions about the system and its uncertainty in order to be able to perform the training in the offline setting. This is also what we did above, by taking certain assumptions about the building domain and even taking real, historic data into account. However, while this may mean that the reinforcement learning enabler learns a policy that solves this specific problem (i.e., under the given assumptions), the learned policy can be useless or may even perform worse than a policy only trained online when applied to the actual problem at run-time (which may violate these assumptions), even if it is relatively similar. One approach to this problem is to leverage the emerging concept of deep meta reinforcement learning.

Coping with large discrete action spaces Existing online reinforcement learning solutions for self-adaptive services propose randomly selecting adaptation actions for exploration he effectiveness of exploration therefore directly depends on the size of the adaptation space, because each adaptation action has an equal chance of being selected. Some reinforcement learning algorithms can cope with a large space of actions, but require that the space of actions is continuous in order to generalize over unseen actions. Self-adaptive Smart IoT Systems may have large, discrete adaptation spaces; *e.g.* if their adaptations entail reconfigurations of many system features or a large set of discrete parameters. In the presence of such large, discrete adaptation space, random exploration thus may lead to slow learning at run-time. One approach to this problem is to leverage the structure of the adaptation space to better guide the exploration process. In related work, we have demonstrated that such improved exploration is possible for cloud services [13]. It thus can serve as a promising basis for applying to Smart IoT Systems.

References

[1] Mehdi Amoui *et al.* “Adaptive action selection in autonomic software using reinforcement learning”. In: *Fourth International Conference on Autonomic and Autonomous Systems (ICAS’08)*. IEEE. 2008, pp. 175–181.

[2] Hamid Arabnejad *et al.* “A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling”. In: *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE. 2017, pp. 64–73.

[3] Rafael Aschoff and Andrea Zisman. “Qos-driven proactive adaptation of service composition”. In: *International Conference on Service-Oriented Computing*. Springer. 2011, pp. 421–435.

[4] Enda Barrett, Enda Howley, and Jim Duggan. “Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. In: *Concurrency and Computation: Practice and Experience* 25.12 (2013), pp. 1656–1674.

[5] Enda Barrett and Stephen Linder. “Autonomous hvac control, a reinforcement learning approach”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2015, pp. 3–19.

[6] Tao Chen and Rami Bahsoon. “Self-adaptive and online qos modeling for cloud-based software services”. In: *IEEE Transactions on Software Engineering* 43.5 (2016), pp. 453–475.

[7] Nicolas D’Ippolito *et al.* “Hope for the best, prepare for the worst: multi-tier control for adaptive systems”. In: *Proceedings of the 36th International Conference on Software Engineering*. 2014, pp. 688–699.

[8] Xavier Dutreilh *et al.* “Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow”. In: *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*. 2011, pp. 67–74.

[9] Didac Gil de la Iglesia and Danny Weyns. “MAPE-K Formal Templates to Rigorously Design Behaviors for Self-Adaptive Systems”. In: *TAAS* 10..3 (2015), 15:1–15:31.

[10] Pooyan Jamshidi *et al.* “Machine learning meets quantitative planning: Enabling self-adaptation in autonomous robots”. In: *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, IEEE. 2019, pp. 39–50.

[11] Jeffrey O. Kephart and David M. Chess. “The Vision of Autonomic Computing”. In: *IEEE Computer* 36.1 (2003), pp. 41–50.

[12] Tania Lorido-Botran, Jose Miguel-Alonso, and Jose A Lozano. “A review of auto-scaling techniques for elastic applications in cloud environments”. In: *Journal of grid computing* 12.4 (2014), pp. 559–592.

[13] Andreas Metzger *et al.* “Feature Model-Guided Online Reinforcement Learning for Self-Adaptive Services”.In: *Service-Oriented Computing – 18th International Conference, ICSOC 2020, Dubai, United Arab Emirates, December 14–17, 2020, Proceedings*. Ed. by Eleanna Kafeza *et al.* Vol. 12571. Lecture Notes in Computer Science. Springer, 2020, pp. 269–286. DOI: 10.1007/978-3-030-65310-1_20. URL: https://doi.org/10.1007/978-3-030-65310-1%5C_20.

[14] Ahmed Moustafa and Minjie Zhang. "Learning efficient compositions for qos-aware service provisioning". In: *2014 IEEE International Conference on Web Services*. IEEE. 2014, pp. 185–192.

[15] Ofir Nachum *et al.* "Bridging the Gap Between Value and Policy Based Reinforcement Learning". In: *Advances in Neural Information Processing Systems 12 (NIPS 2017)*. 2017, pp. 2772–2782.

[16] Alexander Palm, Andreas Metzger, and Klaus Pohl. "Online reinforcement learning for self-adaptive information systems". In: *International Conference on Advanced Information Systems Engineering*. Springer. 2020, pp. 169–184.

[17] Andres J. Ramirez, Adam C. Jensen, and Betty H.C. Cheng. "A taxonomy of uncertainty for dynamically adaptive systems". In: *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE. 2012, pp. 99–108.

[18] Mazeiar Salehie and Ladan Tahvildari. "Self-adaptive software: Landscape and research challenges". In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 4.2 (2009), pp. 1–42.

[19] John Schulman *et al.* "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[20] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[21] Richard S. Sutton *et al.* "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Advances in Neural Information Processing Systems 12 (NIPS 1999)*. 2000, pp. 1057–1063.

[22] Gerald Tesauro *et al.* "On the use of hybrid reinforcement learning for autonomic resource allocation". In: *Cluster Computing* 10.3 (2007), pp. 287–299.

[23] Hongbign Wang *et al.* "Integrating reinforcement learning with multi-agent techniques for adaptive service composition". In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 12.2 (2017), pp. 1–42.

[24] Tianqi Zhao *et al.* "A reinforcement learning-based framework for the generation and evolution of adaptation rules". In: *2017 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE. 2017, pp. 103–112.

DOI: 10.1561/9781680838251.ch7

Chapter 7

Security of Smart IoT Systems

By Erkuden Rios, Eider Iturbe, Angel Rego, Saturnino Martinez, Anne Gallon, Christophe Guionneau and Arezki Slimani

7.1 Introduction

Ensuring data confidentiality, integrity, and availability while it is being processed, stored and transmitted by all parts of the environment are high-priority concerns in SIS. One of the most interesting approaches to ensure secure behaviour of the SIS is to embed security features such as monitoring, access control, encryption capabilities, etc. into the IoT Platform used as middleware to capture sensors' data and act as gateway to actuators. As SOFIA-SMOOL, or for short, SMOOL [9], is the IoT platform used in the ENACT Smart Building use case (see Chapter 11), the project has worked in extending this platform with built-in features that enable the platform to implement some of the required security controls to prevent integrity, confidentiality, access control and non-repudiation related issues. Chapter 7.2 describes how the SMOOL platform can be used in the SIS development and operation to monitor and control the desired security properties in the access to resources and communications between smart things of the SIS.

Security assurance at operation does require an external service, agnostic to system design but tailored to final system deployment, that is supervising at all times the security behaviour of the different elements in the IoT system. The role of this security monitoring service is to make sure that security incidents or anomalies are early identified and corresponding alerts are raised to system operators. Chapter 7.3 describes the ENACT enabler supporting at operations the situational awareness of SIS, the so called, *Security and Privacy Monitoring Enabler*. The enabler is capable of collecting data from different layers of the IoT system: network, system and application layers. All these data are combined by the tool for advanced intrusion detection and anomaly detection. Artificial intelligence detection mechanisms are combined with a multi-layer surveillance so as accurate information of holistic security status of the SIS and all its parts is enabled.

Last but not least, Chapter 7.4 brings an innovative approach to access control in SIS. The tool implementing it is named *Context Aware Access Control* since it offers context-based authentication and authorisation of devices and services exchanging data within the SIS. The chapter describes the various manners in which this tool can be used to secure the IoT accesses, considering contextual information in form of a dynamic risk level computation. The context-awareness capability of the tool has been integrated and validated in the eHealth use case described in Chapter 9.

7.2 Built-in Security in IoT Platforms

7.2.1 Security-by-Design in IoT Platforms

Complex systems usually cover the security aspects by adding a layer intersecting or covering other business layers (user interface, data management, processes, etc.). When dealing with IoT systems, the heterogeneous nature of sensors, communication channels, Edge devices or Cloud services often demands the architects focusing on business logic, leaving unattended the needed security controls on sensitive areas (e.g. securing the Edge devices, credentials management for key devices,...), even if some sensors may use weak encryption or even produce data in clear because they rely on transport layer encryption.

Therefore, most of the security management is often handled by the developers creating dedicated solutions on the IoT platform. The platform could provide its own battle-tested security mechanisms but those may not fit well or at all with the security features required by the application developers. In these cases, they are impelled to provide additional security measures to the ones available in the IoT platform. And this may bring problems because when custom security is implemented it is likely that flaws occur, particularly when the developer is not a security expert.

We can introduce the security improvements performed in the SMOOL [1, 9] IoT middleware as an example of how adding security features "by design" generates important benefits, like the use of better security patterns, ensuring developer confidence on the global security, and focus efforts on IoT application logic development and testing, rather than on security aspects.

In this example, we will analyse an application IoT client component exclusively. These components are always the weakest part in potential attacks, since they have limited resources to implement security mechanisms. While servers can also be attack targets, they are usually better prepared and include more or more robust security controls, and changes in the servers are always reviewed and tested exhaustively to prevent scalability problems or vulnerabilities. In SMOOL terminology, a KP (Knowledge Processor) is a client communicating with a SIB (Semantic Information Broker) or server. The KP can send sensor data or actuation orders, and it can also subscribe to messages emitted by other clients. The SMOOL KP clients compose all the messages in Smart Space Access Protocol (SSAP) format (particular of SMOOL), where the sensors, the data and the metadata are provided in the semantic W3C's Web Ontology Language (OWL) format. This allows other KPs or clients to subscribe to and consume information concepts in the same way for multiple types of sensors. If two different sources such as a complex industrial machine and a simple ambient sensor are providing temperature data to the SMOOL server, another KP could subscribe through the same mechanism to temperature concept in both source KPs to get the temperature value from each.

When embedding security mechanisms in SMOOL IoT platform, three different approaches can be followed, all of them were tested in ENACT and explained below.

7.2.1.1 Custom code of security controls in the KPs

The first implementation of security features within SMOOL clients consisted in adding security metadata to the business data, that is, for example, adding security metadata to the sensing data transmitted by sensors. For instance, if the client was transmitting temperature data (value, unit, timestamp), we could also attach the type and content of security information. These metadata were added as semantic concepts in SMOOL ontology so as they can be published and subscribed to by KPs just the same as KPs do with business semantic concepts. This way, specialized security KPs could only listen to which security data is flowing, instead of subscribing to all sensor types containing security metadata. The new security concepts added in ENACT to the SMOOL ontology covered authentication, authorization, confidentiality, integrity and non-repudiation. They were created to allow flexibility in the exact implementation of the security (for instance, integrity can allow symmetric or asymmetric key-based payloads). The new security concepts are shown in Figure 7.1 just as they appear in the Protégé application [13] used to visualize the ontology.

This way, the KP developers could create sensor KPs that publish sensor data with security information. Other KPs subscribed to the sensor data would check

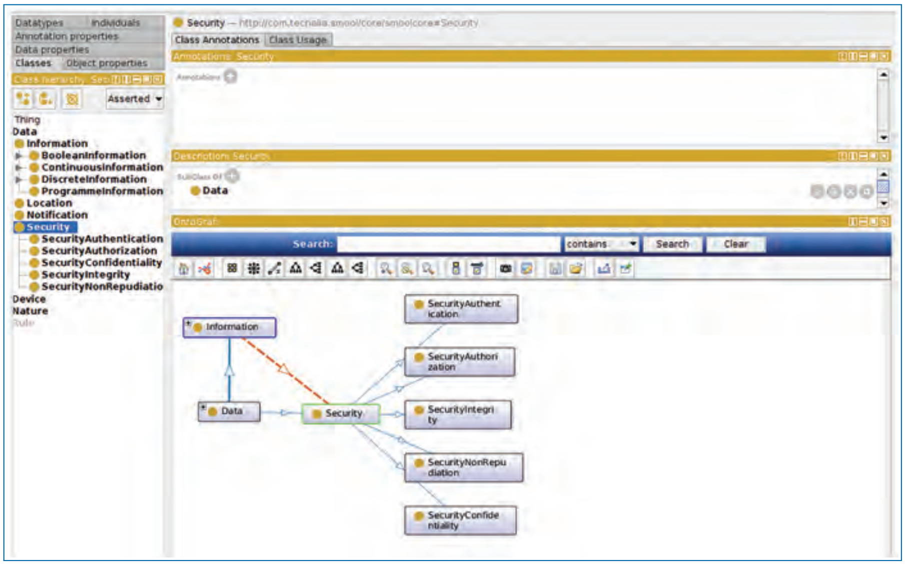


Figure 7.1. SMOOL ontology: Security metadata as ontology concepts.

the security data before accepting any values contained in the message. For instance, the publishers of sensor data could add integrity data, while actuation orders would be sent with valid authorization data.

This approach has several potential failures. The first one is that the developer must add extra code to manage security, which means more lines to peer-review and test, and the application implementation would be prone to insecure execution paths when running. The second problem is that the developer could miss some of the things to double-check when using one of the security concepts in the ontology. For instance, the need of salting before hashing encrypted payloads, or the need to authenticate the device before allowing it to publish data. The third issue is that freedom in security coding increases the list of potential security mistakes a non-expert may make and an expert should review.

In Figure 7.2 a simplified version of KP layers is displayed. When a message containing sensor data (in SSAP format) arrives, the first layer is the Comms layer or communications stack, responsible for accepting and assembling the message by using any of the allowed connectors (TCP, Bluetooth, etc.). The second layer is the Model layer where various operations on the message take place, such as parsing, data insertion into the ontology containers, comparison of previous and updated values, etc. These are the core layers, which facilitate the work of IoT application developers but their drawback is that they are black boxes for them. The next layers are the ones created for the real application, including the Custom code layer to collect, send or retrieve sensing data, and the Security layer with the security code. The figure shows these two layers separated logically, although in reality they

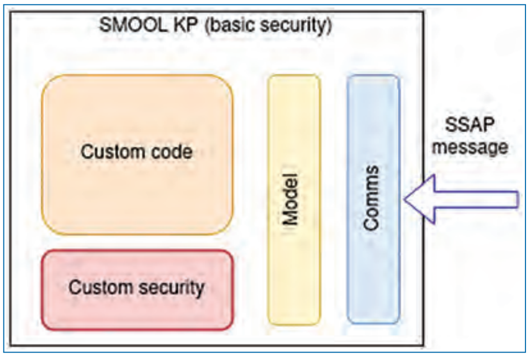


Figure 7.2. SMOOL KP (client) layers. Security is handled by the KP developer.

are glued together. The Security layer deals with the management of the security elements. Since these elements are also part of the ontology, in ENACT we have developed a user-friendly API for creating content of these elements.

7.2.1.2 Basic SecurityChecker in the core of the KPs

In the second approach to security for KPs, the aim was to provide a better experience for the KP developer by providing basic built-in security from the KP design, and therefore, preventing potential security flaws introduced by inexperienced developers. Security policy usage philosophy was added to the KP and implemented in the KP core layers. The security control is performed by a SecurityChecker class added in the security layer, which works on all messages received by a KP extracting any security concept present in the message and testing it against a list of policies. If the message does not conform to the policies, the message will be rejected directly from the core layer, so custom code layer will never be aware that the message was received. This solution is more efficient because there is an automatic security check installed on every newly generated KP, since the mechanism comes in the KP design itself. The developer has also fewer lines of security code to implement, because, instead of needing to program the checks of every message for different security constraints fulfillment, some simple one-line rules are defined as policies. For instance, all actuation orders to a specific actuator type (e.g. blinds in a smart building) must contain authorization metadata.

In summary, this second approach, depicted in Figure 7.3, introduces two major differences compared to the previous approach in Figure 7.2. First, the security layer is now part of the core layers, shown as a single vertical layer that works for all the messages, prior to the custom code execution. Second, custom security code is smaller in number of lines, because it would only be dedicated to the enforcement of advanced security features, such as the management of sensitive data, while most of the messages will be filtered or passed by the core security layer.

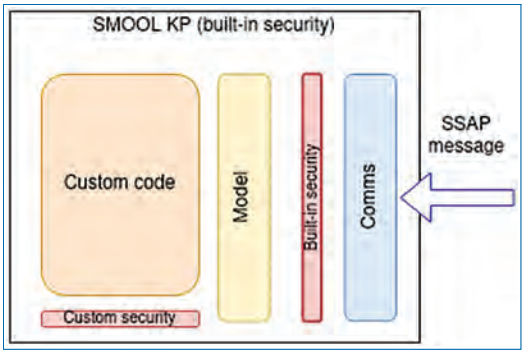


Figure 7.3. SMOOL KP (client) layers. Security is enforced for all messages.

7.2.1.3 External SecurityChecker called from the core of the KPs

The third iteration of built-in security in SMOOL IoT KPs takes advantage of the possibility to use enhanced or external security elements to enforce the needed security policies at all times. This way, the security policies would be completely independent from the KPs and adjustable when needed. The approach extends the built-in SecurityChecker of the KPs to provide better security controls. And these controls are still performed in the core layer, in the same manner as in the second approach. When designing the application, these elements are added as a dependency to replace the standard built-in SecurityChecker. The enhanced controls will be loaded when starting the KP.

To demonstrate this, we have used GeneSIS for deploying refined security controls from the design phase. Since SMOOL and GeneSIS were integrated in ENACT to allow deploying KPs with application extended features, we can also add security features to run either improved extensions of the KP security core layer, or custom security code. Depending on the security needs of the target IoT system, GeneSIS will deploy a different implementation of the security policies, but from the design point of view, the declaration of the security enforcement of the policies is the same regardless what checks the external SecurityChecker will do. In Figure 7.4 below, the security core layer is bigger in lines of code. But for the KP developer the security complexity is the same as in the previous iteration.

Now, the core security box is bigger; however, the knowledge about how security is working in our system remains the same, thanks to the use of policies as main concept. Information reaching the custom code can be treated as secure, for all new security upgrades. The security schema remains straightforward, and the application can keep growing by focusing on the business logic features of the KPs (the custom code layer box) rather than security concerns that are handled outside of it.

Therefore, embracing security features of IoT environments from the design phase carries a set of benefits. The most important benefit is that the majority

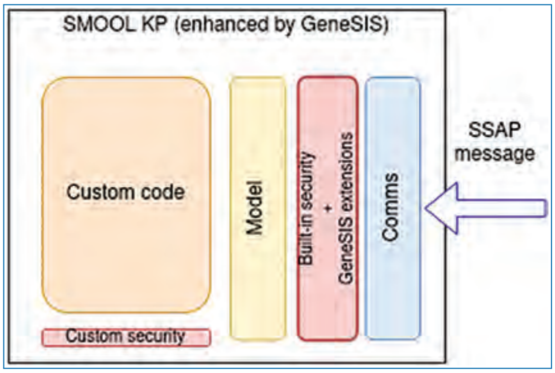

SMOOL KP (enhanced by GeneSIS)
Custom code
Custom security
Model
Built-in security + GeneSIS extensions
Comms
SSAP message

Figure 7.4. SMOOL KP (client) layers. Security is enhanced by GeneSIS redeployment.

of the security flaws can be avoided even when building the very first prototype. The software design can provide some security elements as mandatory and block non-secure messages, and all of it in a user-friendly manner. The second benefit is that developers can trust security elements provided from the design phase instead of needing to develop security features based on ad-hoc preferences. The application to be deployed will be more secure and it would be easier to make new releases in the future. Trends in IoT show that security remains as a main concern, and ENACT has demonstrated that IoT applications trusting on security-aware IoT platforms such as SMOOL can be created in a secure manner to prevent most known issues (legacy libraries or software pieces containing vulnerabilities, broken encryption mechanisms, credentials leakage, etc.). This way, securing applications during the design phase can prevent unexpected risky situations when deploying IoT solutions to production environments.

7.2.2 Reaction to Cyber Incidents and Anomalies

Smart IoT systems allow devices and data generated to be in the core of the system behaviour, leaving human interaction as trigger elements or passive receptors of actions. In SIS most of the elements must run autonomously, re-adapt based on rules, start and stop things, etc. In fact, the SIS behave as complex ecosystems where elements can have different degrees of intelligence but all of them share a high degree of autonomy. And in these systems, a preventive control monitoring what is happening in terms of security is important, but also a reactive control when things go wrong, i.e. issues are detected. For example, hacking only one of the devices in a SIS could create dangerous situations. Imagine a hacked temperature sensor sending low values to keep a heat system running all the time. Now, imagine a hack of a gas or smoke sensor to forge the sensed dangerous values and prevent them from being detected.

Therefore, apart from monitoring and identifying potential issues and attacks, SIS security must be reactive to take countermeasures in real time. The best way to ensure control is having administration rights on the smart devices and Edge, but not all IoT elements can be controlled (for instance, generic sensors or devices from external vendors ready for plug-and-run). Thus, control must sense the IoT system and must act on it, and in cases where the device cannot be managed from the inside, the control must be done from some other part of the communication or data processing chain.

Some security control systems are ready to detect general problems and react to them. Imagine a new device joins a weak security wireless network. This device could start flooding the communications in the network, creating a denial of service for every other legitimate device. The security control system can detect and block that element, no matter which type of device it is.

Now, a more intelligent and refined malicious IoT device could connect to the same security weak network and send legitimate data shaped in the same format other devices are using in their transmissions. The device is accepted, and the data is also accepted because it fits the format expected to be processed. In this case, a smart control element should understand operational data, so as to be able to detect abnormal values and provide feedback to the Security control system.

In the previous Chapter 7.2, we saw how smart IoT devices could enforce security controls based on policies. The enforcement was done inside the device itself. But not all issues could be detected in the device, and security updates may not be available once deployed. For the security issues not detected and blocked from the IoT devices, we need reactive security mechanisms dealing with them. In ENACT he have developed a reactive security control system that relies on SMOOL platform and its clients as explained below.

Let's go back to the SMOOL IoT platform we described in the previous Chapter 7.2. The clients or KPs connected to the IoT platform exchange complex messages. Some security features are already handled by the core security layer embedded in every KP, and some other security issues are handled by the generic security control system that reacts to incidents notified by the monitoring system described in Chapter 7.3. However, to enforce security in the communications of the SMOOL ecosystem (the server and the smart things connected to it) we have created a Security KP to control what information is really exchanged in the messages of the other KPs, detect non-secure elements, notify the incident, and block the insecure elements.

Figure 7.5 depicts an attack performed by an elaborated rogue application disguised as an IoT client, i.e. a SMOOL KP. The malicious KP behaves as a normal KP so connection to server and message exchange is allowed. Note that other type of rogue KPs will be rejected if their message structure does not conform to the one

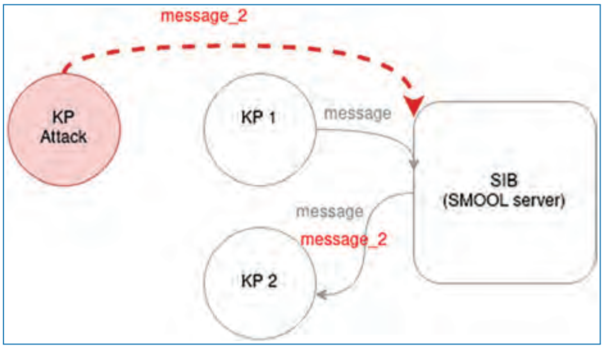


Figure 7.5. SMOOL. Malicious application disguised as a KP.

required by the IoT platform. Thus, this attacking KP is a real danger in the system and cannot be detected by the security infrastructure. The client KP2 receives messages from benign KP1 but also malicious messages from KP Attack (in red).

To solve this problem, a Security KP was created which is a special SMOOL client that has the unique ability to access and understand every SMOOL message. The Security KP, being a client rather than a library in the SMOOL server, has another characteristic: it can be upgraded with new features or customized controls faster than if it was allocated inside the server.

Instead of subscribing to all kinds of SMOOL messages and all the ontology concepts, the Security KP can subscribe to a subset of concepts corresponding to those security properties it needs to handle reactions for. Since security metadata was added in the ontology in the same manner as business data, the Security KP can process all or part of the messages to produce faster reactive responses. The first filter could be to check if messages are following the security polices, then inspecting the actual security metadata, and finally, looking for anomalies in the logic or operational data. If any unwanted message is detected, the Security KP has the right credentials to invoke the global Security control system to analyse the metadata or request it to block all communications from the device generating these messages.

Since the IoT server is the real link to the insecure device, a minor implementation for blocking KPs has been developed. This action can be performed only by the global Security control system. Figure 7.6 illustrates how the Security KP can detect refined attacks.

This time, even if the security metadata sent by the malicious KP Attack was fine (and therefore, the KP2 did not reject the message), the other metadata of the message were not compliant with the refined security rule set in the Security KP, so this KP requests the Security control system to block the KP Attack. The Security control system orders SMOOL server to cut off the connection for the offending KP, and add the offending IP to the black-list. The attacker KP would not be able to send further messages because it will not be even able to connect to the server.

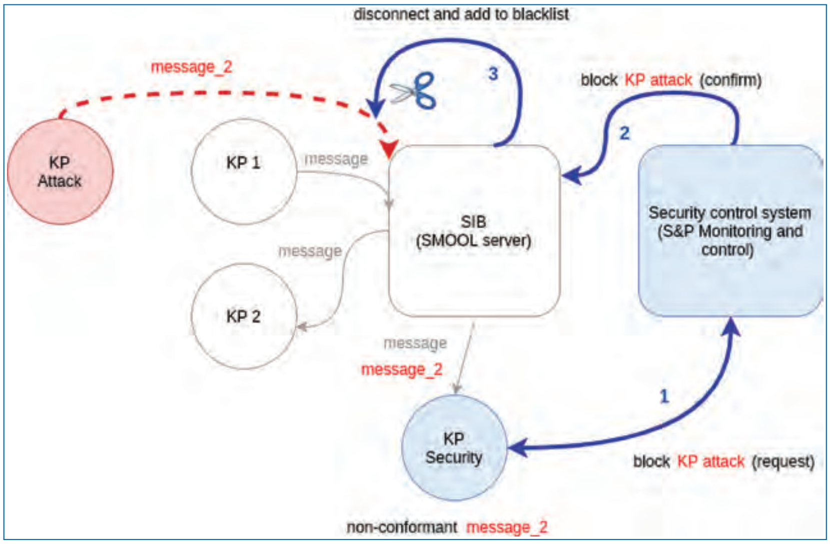

disconnect and add to blacklist
message_2
3
block KP attack (confirm)
2
KP Attack
KP 1
message
SIB (SMOOL server)
Security control system (S&P Monitoring and control)
message
KP 2
message
message_2
1
KP Security
block KP attack (request)
non-conformant message_2

Figure 7.6. SMOOL. Malicious KP detected by Security KP.

By using the Security KP, the application-level messages can be tested against a detailed rule set, the IoT data can be transformed into another format that the Security control system could parse, and problems can be detected and blocking orders invoked too. The application can also detect even more sophisticated attacks depending on the use case, because it could have an additional white-list of allowed IoT devices based on historical activity, or detect abnormal behaviour values from legitimate devices, and then request the Security control system to inspect the device.

7.3 Continuous Monitoring and Detection in IoT System Operation

Information Security Continuous Monitoring (ISCM) is defined by NIST as *"maintaining ongoing awareness of information security, vulnerabilities, and threats to support organizational risk management decisions"* [7]. In an IoT environment, implementing ISCM provides the security administrator of the SIS with means for continuous situational awareness of the cybersecurity and privacy status of the system. This resource supports the security administrator by identifying cybersecurity incidents along with the targeted assets, as well as informing about the criticality and importance of those incidents so that the security expert can decide on the best cybersecurity strategy to mitigate the cyber threats and protect the assets.

In order to better comprehend the importance of the continuous security monitoring, a review of the NIST Security and Privacy Controls for Information Systems and Organizations (SP 800-53 Rev. 5) [8] gives the following quick conclusion: at least 68 security controls of the catalogue are explicitly associated with the monitoring activity distributed in 9 control families: *Access Control, Audit and Accountability, Assessment, Authorization, and Monitoring, Configuration Management, Incident Response, Physical and Environmental Protection, Program Management, Risk Assessment, System and Communications Protection, and System and Information Integrity.*

The standard ISO/IEC 27035 identifies multiple technologies as sources of the required security information and events of continuous monitoring as part of detection and reporting phase within the security incident management process [12]. Mentioned technologies include: Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), honeypots, log monitoring systems, security information and event management systems, and network monitoring systems, among others.

Implementing and deploying ISCM mechanisms into IoT environments may become a complicated task due to the high heterogeneity of standards, technologies, protocols and deployment architectures in use. Despite of the complexity, trust models based on security and privacy technologies deployed in IoT systems will be more and more necessary to ensure consumer acceptance [6].

7.3.1 Architecture and Main Capabilities

In ENACT, ISCM area is covered by the Security and Privacy Monitoring and Control Enabler (S&P Mon&Con), which aids the SIS operator in learning at all times the security status of the SIS and control the behaviour of the SIS in order to ensure it adheres to the security requirements designed. This Enabler delivers three main capabilities:

- Flexible and extensible continuous monitoring mechanism. A comprehensive security monitoring involves having granular, modular and dynamic security controls to be coordinated with. In this way, the enabler can be adapted to work properly with different types of security controls as data sources, and furthermore, the enabler can even be configured to respond through the use of specific security controls deployed in the SIS itself.
- Advanced anomaly detection through user and entity behaviour analysis using Artificial Intelligence (AI) techniques. Based on a zero trust security model, the enabler follows a cybersecurity strategy of addressing both internal and external threats. Particularly, all internal users and entities are considered for the anomaly-based Intrusion Detection System analytics.
- Scalability of the solution. Both the modular architecture and the technologies the enabler is based on guarantee the solution is able to rapidly scale up in large-scale IoT system scenarios, which also implies the need to deploy

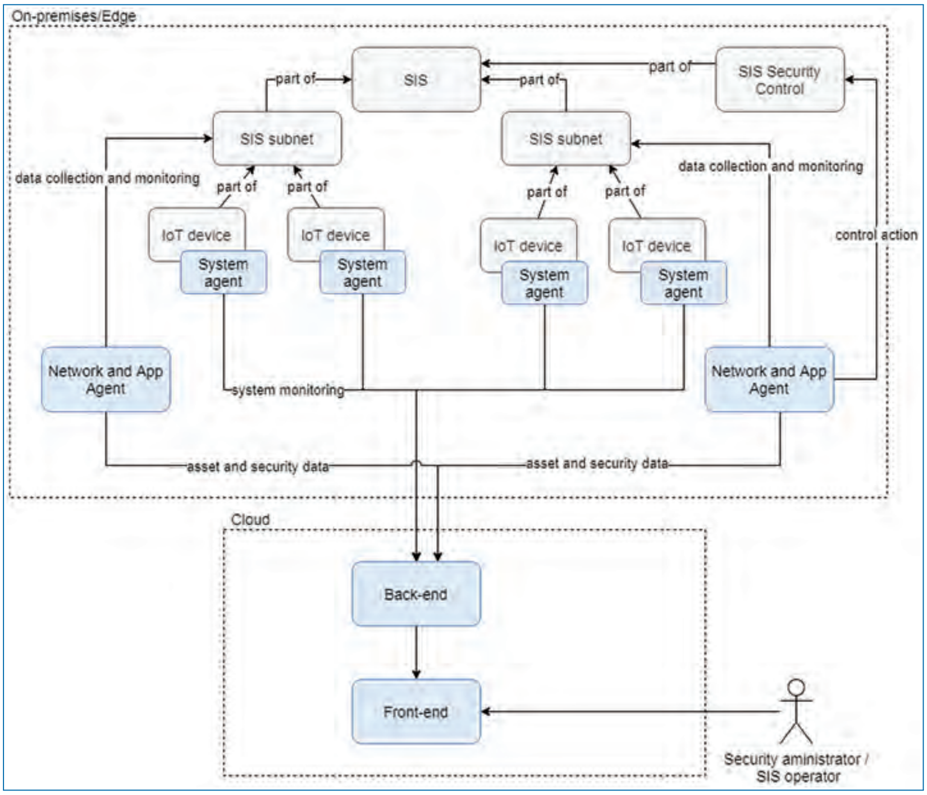


Figure 7.7. High-level architecture of the Security and Privacy Monitoring and Control Enabler.

hundreds or thousands of monitoring agents depending on the extension of the system.

Figure 7.7 shows the high-level architecture of the Security and Privacy Monitoring and Control Enabler. The enabler captures different kinds of data from the SIS through multiple distributed probes named *agents*. There are three types of monitoring agents:

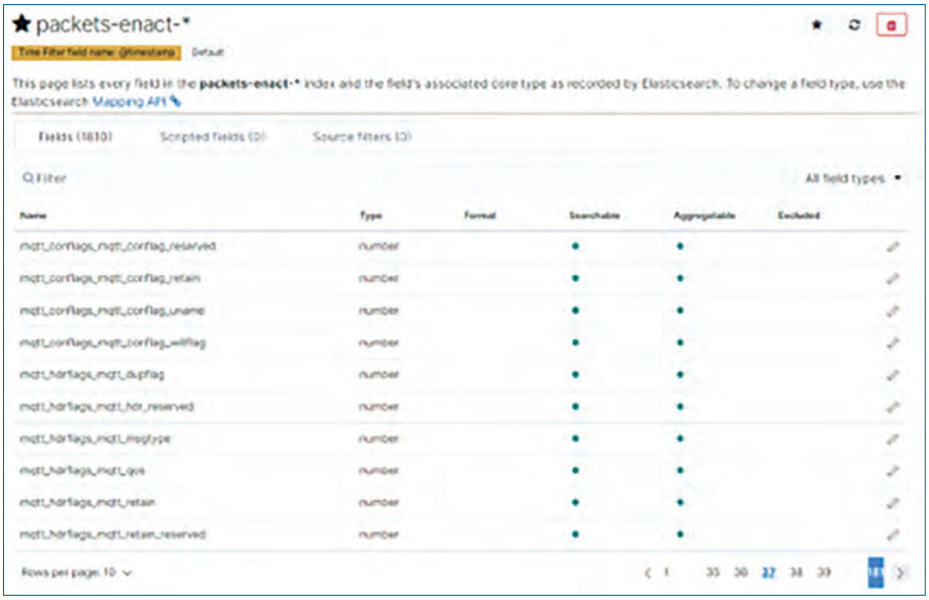
1. *Network agent*, which captures network traffic data, network related security events and asset related data. It integrates an open source signature-based network IDS that is able to detect well-known security attacks and generate security events accordingly. Besides that, the agent includes capabilities to generate protocol-specific security events, e.g. related to ARP protocol so as to enable the detection of ARP spoofing attacks.
2. *System agent*, which gathers log information and data related to the activities and processes within the devices of the SIS; and,

3. *App agent*, which collects data at application layer and generates security events accordingly. This agent can be customized depending on the SIS characteristics; e.g. if the SIS is in intensive use of the MQTT protocol, this agent can be developed with specific MQTT based rules for monitoring and controlling that only authorized users and assets (smart things, devices, services, etc.) can communicate in the SIS.

All the data gathered by the monitoring agents is sent to the back-end of the enabler. Depending on the size of the SIS, the amount of data recovered after some time can be huge which would likely cause processing difficulties. In order to avoid a bottleneck at next phases of data processing and analytics, the entry of the back-end is implemented by a streaming bus (based on the open source distributed streaming platform Apache Kafka [4]. Moreover, the streaming bus provides extensibility to the solution by offering multiple channels where various kinds of data can be collected, and it also allows exchanging the outcome from the enabler in form of identified security events with external components.

The back-end of the enabler includes a data storage infrastructure which stores in a NoSQL database (based on Elasticsearch [2]) all the acquired and pre-processed data from the agents. Multiple indexes are created in the storage infrastructure depending on the different sources of data. For example, in a Smart Building IoT system where many communication protocols can be working at the same time, the network data can easily be stored with the definition of approximately 1800 network attributes as shown in Figure 7.8. Bearing in mind that network traffic is only one of the multiple data sources considered for the analytics within the enabler, dealing with enormous amounts of heterogeneous data is one of the major challenges addressed by this enabler in IoT environments.

The main ground-breaking part of the enabler is the anomaly detection service included in the back-end. AI techniques have been leveraged to analyse the collected SIS data and security events in order to detect anomalies in the system. Mainly, unsupervised Deep Learning techniques have been used to perform the SIS behavioural analytics and anomaly detection. Many Deep Learning techniques have been studied depending on the SIS characteristics so as to implement an accurate anomaly-based detection system. Figure 7.9 depicts Vanilla Long short-term memory (LSTM), a type of recurrent neural network (RNN) architecture, predictions for MQTT protocol in a Smart Building system showing as red points the anomalies detected correctly, where real MQTT traffic behaviour deviates largely from predicted one.



packets-enact-*

Time Filter field name: @timestamp Default

This page lists every field in the **packets-enact-*** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch Mapping API

Fields (1810) Scripted fields (0) Source filters (0)

Filter All field types

Name	Type	Format	Searchable	Aggregatable	Excluded
mqtt_conflags_mqtt_conflag_reserved	number		●	●	
mqtt_conflags_mqtt_conflag_retain	number		●	●	
mqtt_conflags_mqtt_conflag_uname	number		●	●	
mqtt_conflags_mqtt_conflag_willflag	number		●	●	
mqtt_hdrflags_mqtt_dupflag	number		●	●	
mqtt_hdrflags_mqtt_hdr_reserved	number		●	●	
mqtt_hdrflags_mqtt_msgtype	number		●	●	
mqtt_hdrflags_mqtt_qos	number		●	●	
mqtt_hdrflags_mqtt_retain	number		●	●	
mqtt_hdrflags_mqtt_retain_reserved	number		●	●	

Rows per page: 10

< 1 ... 35 36 37 38 39 ... >

Figure 7.8. Extract of the network fields of stored indexes in the Smart Building.

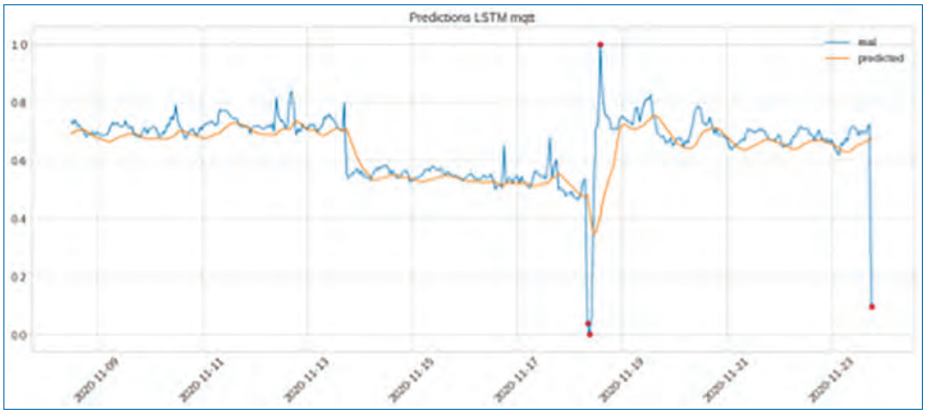


Figure 7.9. Vanilla LSTM predictions for MQTT protocol in the Smart Building.

Considering that each SIS can be completely different in terms of diversity of network protocols in the communications, type and number of devices, type and amount of operational data exchanged, etc. the anomaly detection capability of the enabler must be adjusted to each type of SIS, which means that the detection models need to be re-trained for each SIS.

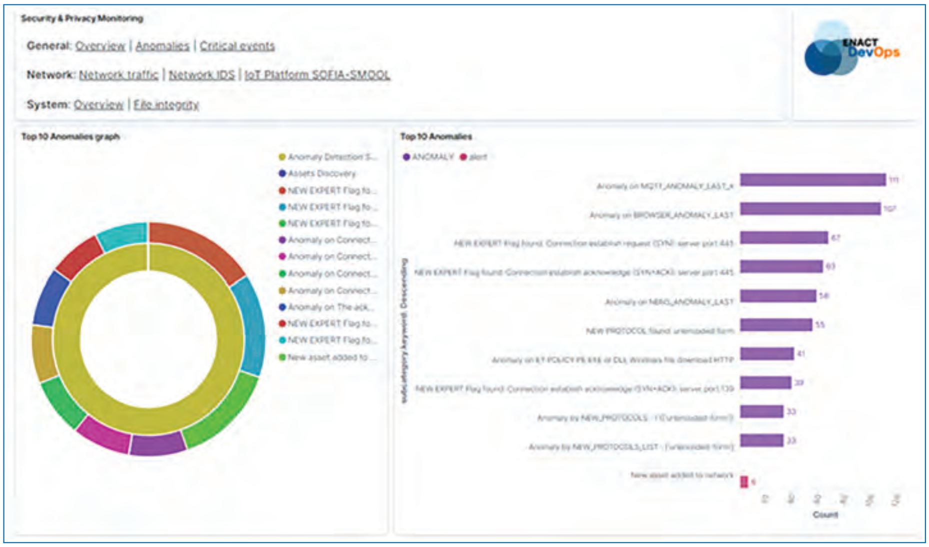

Security & Privacy Monitoring
General: Overview | Anomalies | Critical events
Network: Network traffic | Network IDS | IoT Platform SOFIA-SMOOL
System: Overview | File integrity
ENACT DevOps
Top 10 Anomalies graph
Anomaly Detection S...
Assets Discovery
NEW EXPERT Flag fo...
NEW EXPERT Flag fo...
NEW EXPERT Flag fo...
Anomaly on Connect...
Anomaly on Connect...
Anomaly on Connect...
Anomaly on Connect...
Anomaly on The ack...
NEW EXPERT Flag fo...
NEW EXPERT Flag fo...
New asset added to...
Top 10 Anomalies
ANOMALY
alert
Anomaly on MQTT_ANOMALY_LAST_x
111
Anomaly on BROWSER_ANOMALY_LAST
107
NEW EXPERT Flag found: Connection establish request (SYN): server port 445
67
NEW EXPERT Flag found: Connection establish acknowledge (SYN+ACK): server port 445
63
Anomaly on NEIG_ANOMALY_LAST
58
NEW PROTOCOL found: unencoded form
55
Anomaly on ET POLICY PE EXE or DLL Windows file download HTTP
41
NEW EXPERT Flag found: Connection establish acknowledge (SYN+ACK): server port 139
39
Anomaly by NEW_PROTOCOLS : 1 (unencoded form)
33
Anomaly by NEW_PROTOCOLS_LIST : (unencoded form)
33
New asset added to network
4
subcategory.keyword: Descending
Count

Figure 7.10. Extract of the overview dashboard view in the Smart Building.

The security administrator or SIS operator can continuously be aware of the cybersecurity status of the SIS using the front-end of the enabler, which displays all SIS status data, prediction data and detected security events in a user-friendly manner in form of alerts, statistics and graphs. Within ENACT, multiple view-points have been implemented in the front-end in order to have a comprehensive overview of the SIS security status; additionally, each of the viewpoints includes many dashboards. Nevertheless, the front-end can be adapted ad hoc in case the end user wants to have more details, or customize the graphs and the rest of the visualization objects.

Figure 7.10 shows an extract of the overview dashboard view of the General viewpoint in the enabler implemented for the Smart Building System use case (cf. Chapter 11). It offers the most important information related to the security events generated over the SIS to protect. The end user can navigate through the rest of the dashboards to learn more details about security events.

Figure 7.11 shows the network traffic dashboard of the Network viewpoint and Figure 7.12 shows an extract of the anomalies dashboard of the General view-point. Both dashboards have also been customized for the Smart Building System use case.

The Security and Privacy Monitoring and Control Enabler has been designed with the capability to integrate with an IoT platform, specifically with the SMOOL

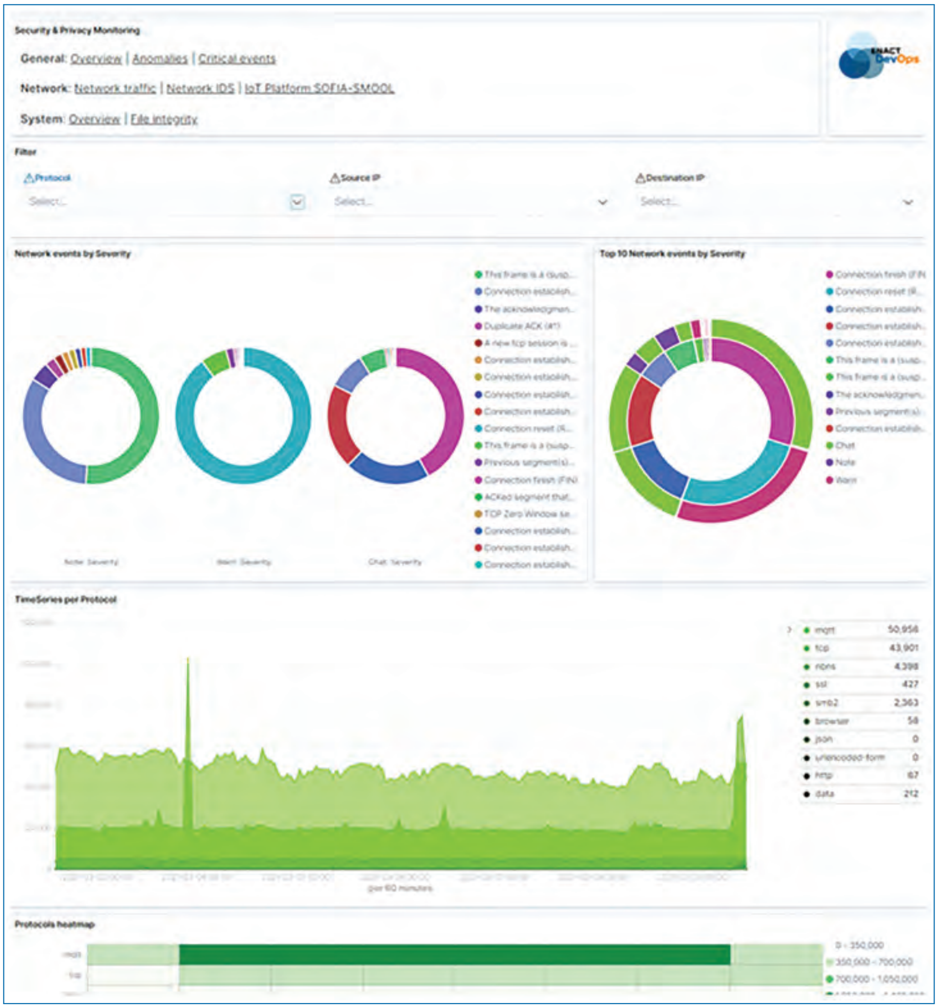
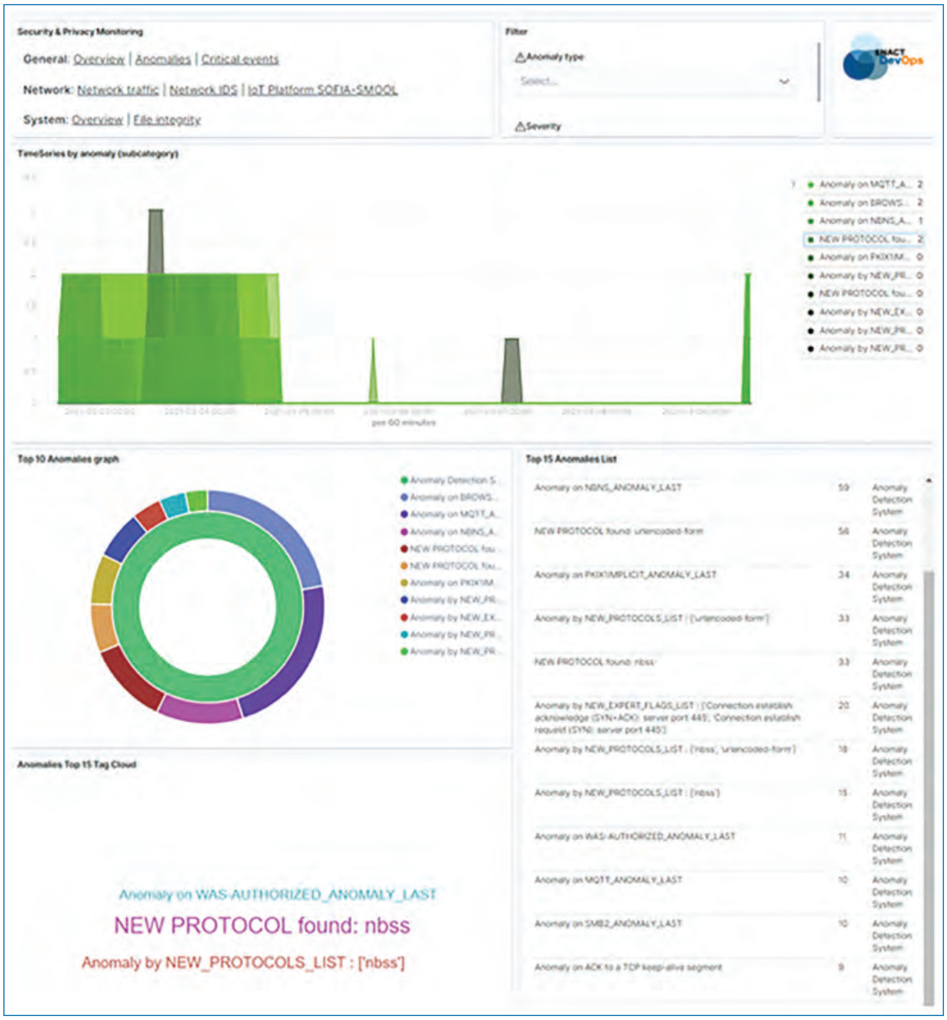

Security & Privacy Monitoring
General: Overview | Anomalies | Critical events
Network: Network traffic | Network IDS | IoT Platform SOFIA-SMOOL
System: Overview | File integrity
ENACT DevOps
Filter
Protocol
Source IP
Destination IP
Network events by Severity
Top 10 Network events by Severity
TimeSeries per Protocol
mqtt 50,956
tcp 43,901
mdns 4,398
ssl 427
smb2 2,363
browser 58
json 0
urlencoded-form 0
http 67
data 212
Protocols heatmap
0 - 350,000
350,000 - 700,000
700,000 - 1,050,000

Figure 7.11. Extract of the network traffic dashboard of the Network viewpoint in the Smart Building.

IoT Platform (cf. Chapter 7.2). This particular implementation allows the security administrator to continuously monitor all communications and data exchanged through the IoT Platform. Figure 7.13 shows the architecture of the enabler integrated with SMOOL IoT Platform.

The distinctive feature in this scenario is that a client of the SMOOL IoT Platform, called Security KP (cf. Chapter 7.2.2), works as an app agent for the enabler by monitoring all data and communications through the IoT Platform and identifying


Security & Privacy Monitoring

General: Overview | Anomalies | Critical events

Network: Network traffic | Network IDS | IoT Platform SOFIA-SMOOL

System: Overview | File integrity

Filter

Anomaly type

Select...

Severity

TimeSeries by anomaly (subcategory)

per 60 minutes

● Anomaly on MQTT_A...	2
● Anomaly on BROWS...	2
● Anomaly on NBNS_A...	1
● NEW PROTOCOL fou...	2
● Anomaly on PKIX1IM...	0
● Anomaly by NEW_PR...	0
● NEW PROTOCOL fou...	0
● Anomaly by NEW_EX...	0
● Anomaly by NEW_PR...	0
● Anomaly by NEW_PR...	0

Top 10 Anomalies graph

● Anomaly Detection S...
● Anomaly on BROWS...
● Anomaly on MQTT_A...
● Anomaly on NBNS_A...
● NEW PROTOCOL fou...
● NEW PROTOCOL fou...
● Anomaly on PKIX1IM...
● Anomaly by NEW_PR...
● Anomaly by NEW_EX...
● Anomaly by NEW_PR...
● Anomaly by NEW_PR...

Top 15 Anomalies List

Anomaly on NBNS_ANOMALY_LAST	59	Anomaly Detection System
NEW PROTOCOL found: urlencoded-form	56	Anomaly Detection System
Anomaly on PKIX1IMPLICIT_ANOMALY_LAST	34	Anomaly Detection System
Anomaly by NEW_PROTOCOLS_LIST : ['urlencoded-form']	33	Anomaly Detection System
NEW PROTOCOL found: nbss	33	Anomaly Detection System
Anomaly by NEW_EXPERT_FLAGS_LIST : ['Connection establish acknowledge (SYN+ACK): server port 445', 'Connection establish request (SYN): server port 445']	20	Anomaly Detection System
Anomaly by NEW_PROTOCOLS_LIST : ['nbss', 'urlencoded-form']	18	Anomaly Detection System
Anomaly by NEW_PROTOCOLS_LIST : ['nbss']	15	Anomaly Detection System
Anomaly on WAS-AUTHORIZED_ANOMALY_LAST	11	Anomaly Detection System
Anomaly on MQTT_ANOMALY_LAST	10	Anomaly Detection System
Anomaly on SMB2_ANOMALY_LAST	10	Anomaly Detection System
Anomaly on ACK to a TCP keep-alive segment	9	Anomaly Detection System

Anomalies Top 15 Tag Cloud

Anomaly on WAS-AUTHORIZED_ANOMALY_LAST

NEW PROTOCOL found: nbss

Anomaly by NEW_PROTOCOLS_LIST : ['nbss']

Figure 7.12. Extract of the anomalies dashboard of the General viewpoint in the Smart Building.

potential malicious intrusions. Furthermore, the integration with SMOOL IoT Platform provides the enabler security control capability to react to security incidents. For example, when detecting an unauthorized communication within SMOOL IoT Platform by the app agent (i.e. the Security KP), the enabler can respond by blocking all communications coming from the unauthorized client. All these security events registered by the SMOOL IoT Platform are shown in the front-end of the enabler.

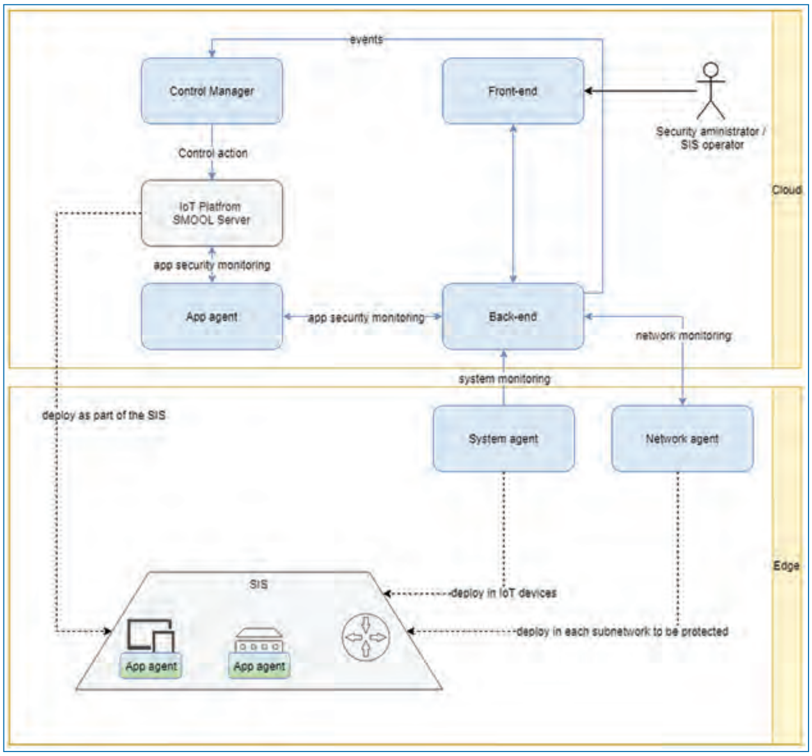


Figure 7.13. Architecture of the the Security and Privacy Monitoring and Control Enabler integrated with SMOOL IoT Platform.

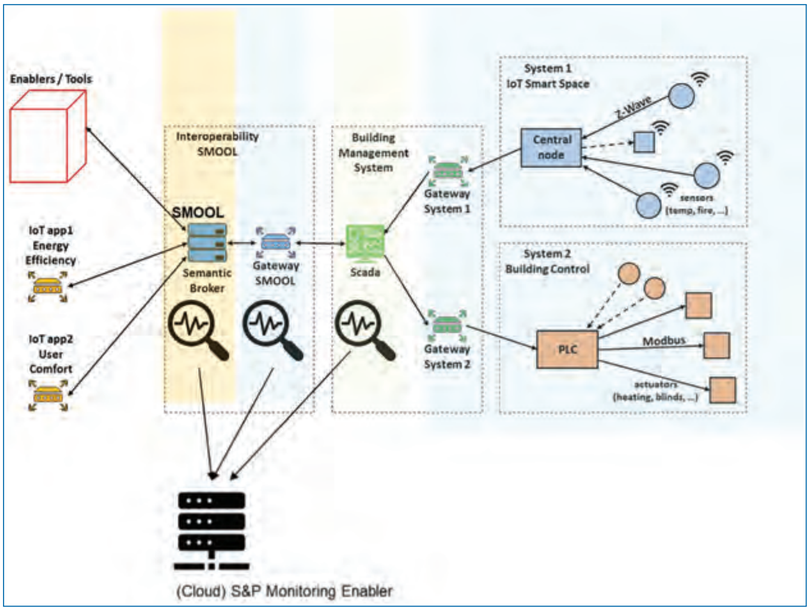


Figure 7.14. Smart Building SIS high-level architecture with the Security and Privacy Monitoring and Control Enabler integrated.

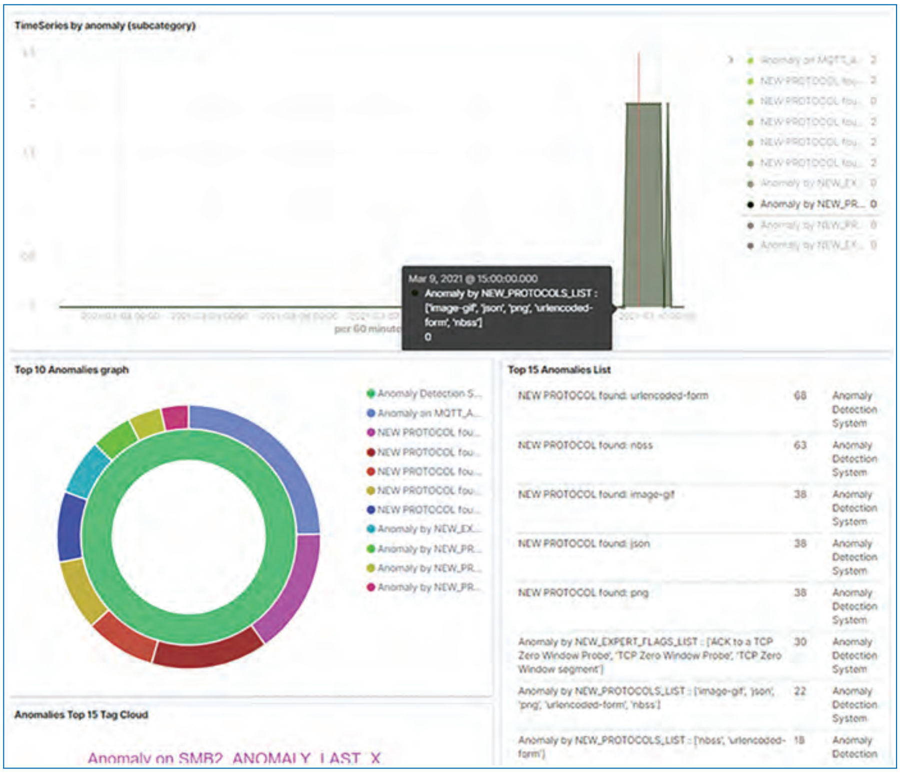
7.3.2 Validation

The Security and Privacy Monitoring and Control Enabler has been validated in two different use cases in the ENACT project.

7.3.2.1 Smart Home System

The Smart Home System (cf. Chapter 11) has integrated the Security and Privacy Monitoring and Control Enabler together with the SMOOL IoT Platform in order to monitor the IoT applications of user comfort and energy efficiency of the building. In that way, the Smart Home System has been monitored at different layers: network layer covered by network monitoring agents, system layer covered by system monitoring agents in IoT devices such as Raspberry Pis, and app layer covered by SMOOL KP clients as app agents.

The SIS operator of the Smart Building System has been able to discover anomalies and security incidents by using the enabler. For example, Figure 7.15 shows an anomaly in the network protocols used by the Smart Building system related to HTTP protocol's content type formats (such as json, image-gif or png); this



Mar 9, 2021 @ 15:00:00.000
Anomaly by NEW_PROTOCOLS_LIST : ['image-gif', 'json', 'png', 'urlencoded-form', 'nbss']
0

Top 15 Anomalies List

NEW PROTOCOL found: urlencoded-form	68	Anomaly Detection System
NEW PROTOCOL found: nbss	63	Anomaly Detection System
NEW PROTOCOL found: image-gif	38	Anomaly Detection System
NEW PROTOCOL found: json	38	Anomaly Detection System
NEW PROTOCOL found: png	38	Anomaly Detection System
Anomaly by NEW_EXPERT_FLAGS_LIST : ['ACK to a TCP Zero Window Probe', 'TCP Zero Window Probe', 'TCP Zero Window segment']	30	Anomaly Detection System
Anomaly by NEW_PROTOCOLS_LIST : ['image-gif', 'json', 'png', 'urlencoded-form', 'nbss']	22	Anomaly Detection System
Anomaly by NEW_PROTOCOLS_LIST : ['nbss', 'urlencoded-form']	18	Anomaly Detection

Figure 7.15. Example of network anomaly detection in the Smart Building.

anomaly can be seen in different graphics of the dashboard: (i) in a timeseries graphic or (ii) in a top 10 anomalies list.

7.3.2.2 Intelligent Transport System

The Intelligent Transport System (cf. Chapter 10) has integrated the Security and Privacy Monitoring and Control Enabler in order to monitor the security status of the on board Edge part of the train system. The enabler has been used and validated for the following scenarios:

1. User and entity behaviour monitoring, which is mainly based on the anomaly detection capabilities offered by the enabler. Industrial protocol network traffic has been analysed in order to detect potential security incidents related to abnormal traffic behaviour. When an anomaly is detected, the enabler is able to react by enabling a specific security control for the SIS itself.
2. Intrusion detection, which uses rule-based detection capabilities implemented within network and app agents of the enabler to spot the unauthorized users and devices trying to communicate or get access to resources in the system.

7.4 Context-aware Access Control

7.4.1 Purpose

Access control and identity governance mechanisms are cornerstones of security and privacy, which is today focused on addressing people accessing IT applications. In the context of the Internet of Things, access control needs to be extended to address not only people accessing IoT, but also to manage the relationships between connected things. This requires designing and building new access control mechanisms for authorizing access to and from connected things, with ad hoc protocols while still being able to address traditional access to IT applications.

The key challenge for access control in IoT is dynamicity. IoT systems are changing continuously: Devices keep entering and exiting the system; The same devices may be used in different context; New connections emerge among the devices; etc. For such highly dynamic IoT systems, access rights from people to devices, and from devices to devices, are not immutable. The access rights may vary according to the context change. Take an eHealth scenario as an example, where senior adults use IoT devices to monitor their physiological data such as blood pressure. In the normal, day-to-day context, only the user himself should have the access right to the data, due to the privacy concern. However, in a special context, such as under emergency rescue, medical staff should be granted with the access right to the journal with historical physiological data. Therefore, the decision of access right in IoT systems must be made with awareness of the context.

The objective of the Context-aware Access Control (CAAC) is to deal with these considerations, by providing dynamic access control mechanisms for IoT systems based on context awareness and risk identification, applicable to both IT (Information Technology) and OT (Operational Technology) domains, through an IAM (Identity and Access Management) gateway for IoT that includes next-generation authorization mechanisms.

Evidian Web Access Manager (WAM) provides security features for identity management and access control. The Context-Aware Access Control tool is an evolution of the authentication and authorization mechanisms provided by WAM intended for the Internet of Things.

7.4.2 Background: Industry Standards of Access Control Protocols

- **The traditional dynamic access control chain based on the XACML model**

  A first approach is to study how the traditional dynamic access control chain based on the XACML model [10] could help to answer the challenge of securing the Internet of Things.

  XACML is a policy-based management system that defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate authorization requests according to the rules defined in policies. As a published standard specification, one of the goals of XACML is to promote common terminology and interoperability between authorization implementations by multiple vendors.

  XACML is primarily an Attribute-Based Access Control (ABAC) system, where attributes associated with an entity are inputs into the decision of whether a given entity may access a given resource and perform a specific action.

  The XACML model supports and encourages the separation of the authorization decision from the point of use. When authorization decisions are baked into client applications, it is very difficult to update the decision criteria when the governing policy changes. When the client is decoupled from the authorization decision, authorization policies can be updated on the fly and affect all clients immediately.

  The access control chain based on the XACML model is depicted in Figure 7.16.

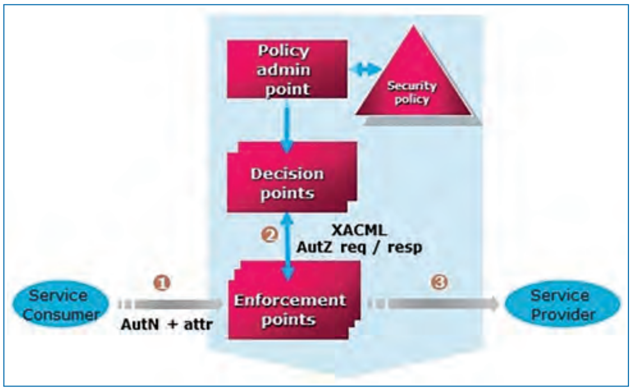


Figure 7.16. The dynamic access control chain based on the XACML model.

In this chain:

- The Policy Decision Point (PDP) evaluates access requests against authorization policies before issuing access decisions.
- The Policy Enforcement Point (PEP) intercepts the user's access request to a resource, makes a decision request to the PDP to obtain the access decision (i.e. access to the resource is approved or rejected), and acts on the received decision.

In fact, this approach is dynamic by essence, since the access control decisions are made based on attributes associated with relevant entities. In addition, it offers a powerful access control language with which to express a wide range of access control policies.

But the following points make this approach prohibitive:

- An approach based on rules is difficult to administer. Defining policies is effort consuming. You need to invest in the identification of the attributes that are relevant to make authorization decisions and mint policies from them. In addition, the ABAC system introduces issues, most notably the 'attribute explosion' [3] issue and, maybe more importantly, the lack of audibility.
- Although Service-Oriented Architecture and Web Services offer advanced flexibility and operability capabilities, they are quite heavy infrastructures that imply significant performance overheads.
- Since XACML has been designed to meet the authorization needs of the monolithic enterprise where all users are managed centrally, this central access control chain is not suitable for cloud computing and distributed system deployment, and it does not scale to the Internet.

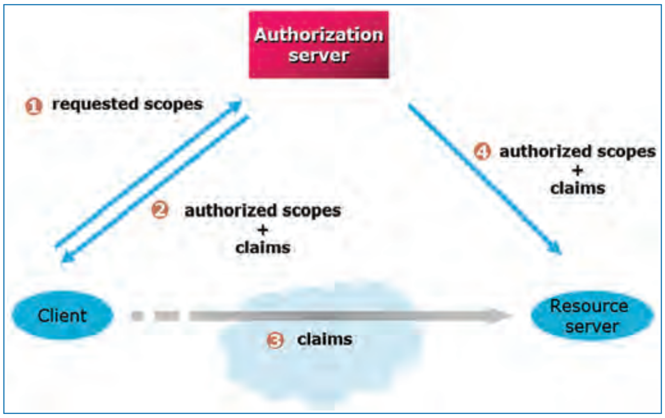


Figure 7.17. Another approach based on OAuth 2.0.

- **Another approach based on OAuth 2.0**

Another approach has been studied, based on the OAuth 2.0 industry-standard protocol for authorization [5].

This approach is depicted in Figure 7.17.

In this approach, a client can access a resource on behalf of a user through an authorization delegation mechanism. This assumes that the user has given his consent for the requested scopes.

As a major advantage, this protocol can be implemented in a light way, by leveraging HTTP and REST-based APIs. In fact, OAuth 2.0 supports the mobile device application endpoint in a lightweight manner. Its simplicity makes it the de-facto choice for mobile and also non-mobile applications. Due to the growing importance of Cloud technologies and APIs, the REST architecture is now heavily favoured.

In addition, OAuth 2.0 allows a fluid integration with role management: OAuth 2.0 scopes can be used to provide role-based authorization.

But this protocol does not have the granularity of XACML in terms of rules. And another point is still an obstacle to meet the need of an IoT context-aware access control: the dynamicity, allowing to take into account the context, is not provided by design.

7.4.3 A Solution for a Context-aware Access Control Approach for IoT

Due to the disadvantages observed on the traditional dynamic access control chain based on the XACML model, it appears that a solution based on OAuth 2.0 is more appropriate.

But to provide an IoT context-aware access control mechanism, the gap must be filled to deliver dynamic authorizations based on context by using the OAuth 2.0 protocol.

Starting from security features for identity management and access control based on the protocols OAuth 2.0 and OpenID Connect (OIDC) [11], the approach is to develop an evolution of these authentication and authorization mechanisms intended for the Internet of Things. Due to the dynamic nature of the data regarding the environment of the connected devices and the persons they belong to, this contextual information must be used to manage and adjust the security mechanisms, i.e. consider contextual information in the identification of the entity requesting access and in the evaluation of the conditions to grant access.

By assessing the applicability of OAuth 2.0, the ENACT IoT context-aware access control leverages it as a key protocol for interoperability, by adding dynamicity to the authorization decisions produced by OAuth 2.0, although this was not originally intended in that protocol. This dynamic capability is in charge of taking contextual information into account and inserting it into authorization decisions.

7.4.4 Architecture

The Context-aware Access Control tool provides an authorization mechanism that issues access tokens to the connected objects after successfully authenticating their owner and obtaining authorization. An access token contains the list of claims and scopes that an authenticated user has consented for this object to access. These scopes and claims are used to restrict accesses to the back-end server APIs to a consented set of resources.

This authorization mechanism may be coupled with contextual information to adapt the access authorizations according to them (for example to make certain information more widely available in some urgent case).

To this objective, the Access Control Tool directly communicates with a Risk Server to make dynamic access controls based on the context information during the authorization phase. For example, it can reject the authorization if the access token is valid while other context information does not respect the authorization policy.

The authorization policy is a set of rules that define whether a user or device must be permitted or denied access to a back-end server. An administrator can control this adjustment and create special authorization rules based on the context data provided.

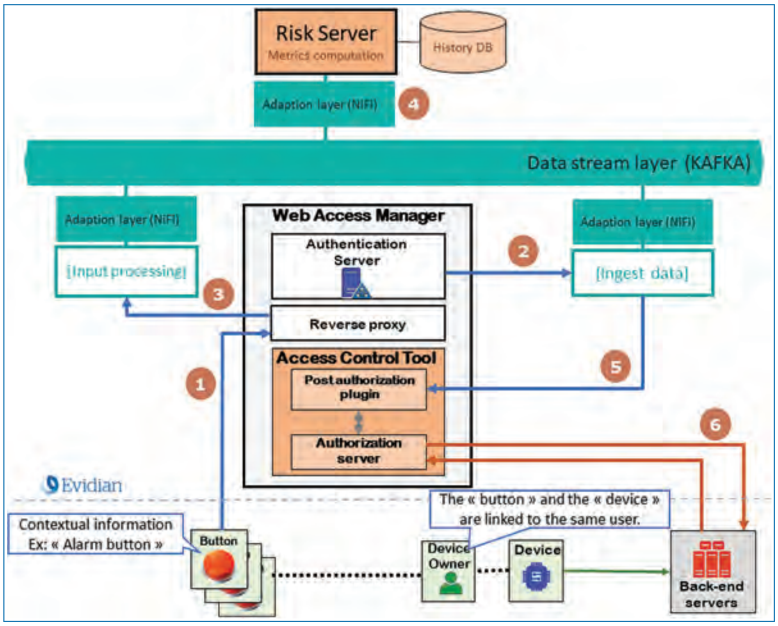


Figure 7.18. Context-aware Access Control global infrastructure.

The Context-aware Access Control tool is provided inside an infrastructure aimed to gather contextual information to deduce a risk level associated with a user. Figure 7.18 gives a global view of this infrastructure.

The infrastructure is based on the Apache Kafka event streaming platform, which allows to publish and subscribe to streams of events. The principle is to publish in this platform contextual information which may come (1) from connected devices (sensors, alarms, etc.) or (2) from audit events produced by the Evidian Web Access Manager. The contextual information is sent to an input processing interface (3) which then publishes it to a Kafka topic. An event is then received by the risk server from an Apache NiFi interface (4) which will take into account the contextual information in a dynamic risk level computation. Then, when a device tries to access a resource, (5) the CAAC retrieves the dynamic risk value associated to the device owner, and (6) this is transmitted to the back-end server to modulate the access accordingly.

In this architecture, two components are providing the Context-aware Access Control mechanisms:

- The **Access Control Tool**, composed of an Authorization server associated to a Post authorization plugin, to add more controls during the authorization phase. Its purpose is to check if the request is authenticated and is authorized to access the back-end server.

Indeed, each time a device sends a request to a back-end server (7), WAM can check the dynamic claims and scopes consented by the user associated to the device that performs the request and, in turn, realize special actions according to this information such as blocking the request or limiting the accessible resources.

The Post authorization plugin extends the basic authorization phase and is entirely customizable. Any operation can be executed during the authorization phase, including calling external programs, and in particular the Risk Server. The Post authorization plugin can create injection variables that can be reused and injected in the initial request sent to the back-end server.

- A **Risk Server** which essentially relies on WAM audit events to calculate a risk level for each user. This allows detection of abnormal behaviour such as connections from unknown IP addresses, or multiple failed connections.

  A user's risk level is a function of the level of trust given to that user. This level of risk determines the level of trust that can be placed in the devices owned by that user. The user's risk level is based on a system of sanctions/rewards depending on the user's behaviour. Its computation uses a ranking system based on a user-specific score: the Risk Score.

  Contextual information coming from external sources (sensors, other applications, etc.) makes it possible to modulate this risk, i.e., to increase or decrease it depending on the situation. For example, in the case of a fire, a smoke detector immediately sends information to the Risk Server, which will considerably reduce the user's risk and allow easier access to resources.

  The contextual information is sent to an input processing interface which then publishes it to a KAFKA topic. For this contextual information transmission to be controlled and secure, the transmitting device must be enrolled in WAM and associated with a user, and it must have received a valid access token which allows it getting its owner's userid to be associated with the contextual information. Each device is associated with a risk factor which can be used to modulate the user's risk score.

7.4.5 Integrating the Context-aware Access Control Tool

- **Device enrolment**

  The device enrolment procedure allows a device to be associated with the identity of its owner. The Access Control tool leverages on the OAuth 2.0 Device Flow protocol to achieve this.

  The only requirements to use this flow are that the device is connected to the Internet and able to make outbound HTTPS requests, and that it is

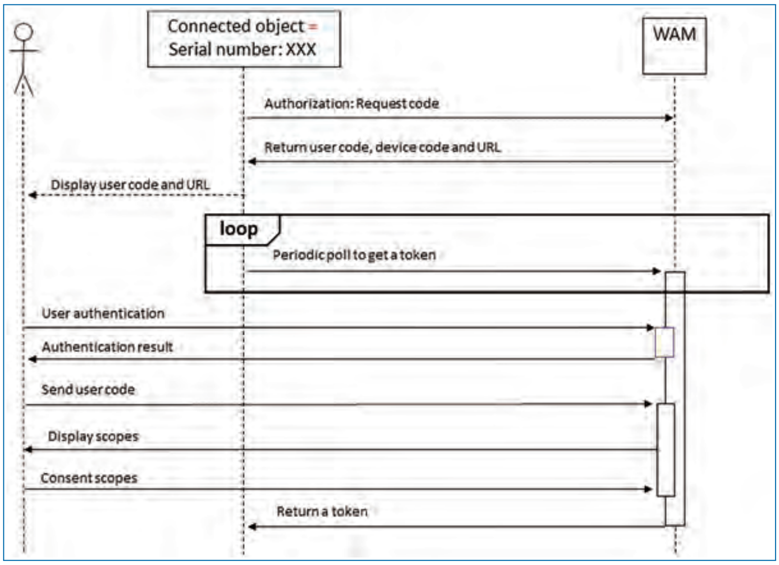

Connected object = Serial number: XXX
WAM
Authorization: Request code
Return user code, device code and URL
Display user code and URL
loop
Periodic poll to get a token
User authentication
Authentication result
Send user code
Display scopes
Consent scopes
Return a token

Figure 7.19. Device enrolment sequence diagram.

able to display or otherwise communicate a URI and code sequence to the user, and that the user (device owner) has a secondary device (e.g., personal computer or smartphone) from which to process the request. There is no requirement for two-way communication between the OAuth client (i.e., the connected device) and the end user's user-agent, enabling a broad range of use-cases.

During this procedure, the user gives his consent to the device to access data scopes on static attributes (username, email, etc.) and also a dynamic attribute (a risk level computed from contextual information on the user). At the end of the enrolment phase, the device receives an access token. The device has now access to the device owner profile that includes static attributes (username, email, etc.) but also the dynamic risk level.

The sequence diagram for this device enrolment procedure is described in Figure 7.19.

- **Context-aware Access Control with WAM used as reverse-proxy**

In this case, WAM is used as a Reverse proxy to protect the back-end servers. Additionally, WAM checks the token of the incoming request to verify if the device is authorized to access the back-end server. If this is the case, WAM injects in the header of the initial request the consent scopes of the device owner. This injection does not modify the request and the scopes injected contain some information about the device owner (username, email, etc..) and a risk level computed from contextual information. This allows the

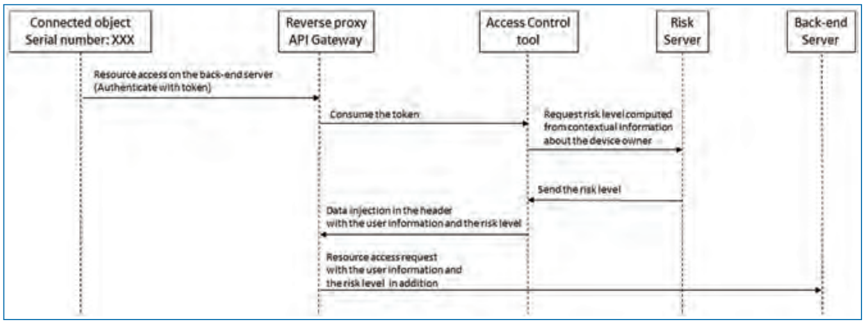

Connected object
Serial number: XXX
Reverse proxy
API Gateway
Access Control
tool
Risk
Server
Back-end
Server
Resource access on the back-end server
(Authenticate with token)
Consume the token
Request risk level computed
from contextual information
about the device owner
Send the risk level
Data injection in the header
with the user information and the risk level
Resource access request
with the user information and
the risk level in addition

Figure 7.20. Context-aware Access Control with WAM as reverse-proxy — Sequence diagram.

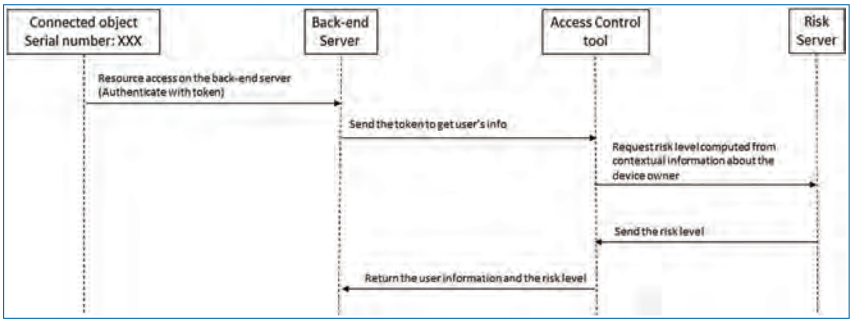

Connected object
Serial number: XXX
Back-end
Server
Access Control
tool
Risk
Server
Resource access on the back-end server
(Authenticate with token)
Send the token to get user's info
Request risk level computed from
contextual information about the
device owner
Send the risk level
Return the user information and the risk level

Figure 7.21. Context-aware Access Control with WAM as OpenID Connect IDP — Sequence diagram.

back-end server to make the link between the requesting device and the user associated with it, and to know the risk level depending on the context.

The sequence diagram for this working mode is described in Figure 7.20.

- **Context-aware Access Control with WAM used as OpenID Connect IDP**

In this case, WAM is used as an OpenID Connect IDP. WAM handles the access control part by checking if the token sent to a back-end server is valid. If this is the case, WAM responds with the consent scopes of the device owner. These scopes contain some information about the user (username, email, etc.) and the contextual risk level.

In the case where the authorization policy is not respected, the Access Control tool will inform the back-end server that it has to reject the request made by the device.

The sequence diagram for this working mode is described in Figure 7.21.

The device uses its token to access the back-end server (for example to push some data). The back-end server checks the validity of the token and retrieves the device owner's consent scopes for this token by calling the user-info endpoint of WAM (the userinfo endpoint from the Access Control tool

API consumes a token to retrieve information on the user). WAM returns the user information (for instance: username, email, address) and the dynamic contextual risk level associated to the user. The back-end applications can use this additional information to perform special actions.

7.4.6 Main Innovation

The main innovation brought by the features offered by the Context-Aware Access Control Enabler can be summarised as follows:

- The solution provides one unique tool 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.
- The solution adds dynamicity to the authorization decisions OAuth 2.0 produces, by injecting dynamic scopes in the standard device flow.
- This allows to exploit contextual risk levels as dynamic attributes in the authorization mechanisms.
- Accordingly, the provided authorizations can be adapted based on a risk level computed from contextual data on the user and his devices, which allows context-aware dynamic access control behaviors.

7.5 Conclusion

This chapter was dedicated to the Security and Privacy Monitoring and Control Enabler designed to be used at the Ops phase of the DevOps life-cycle of SIS to address the security aspects of trustworthy SIS operation. The enabler is an innovative solution that supports SIS operation with multi-source data capturing, advanced detection combining signature-based IDS and AI techniques, and comprehensive situational awareness through a rich multi-viewpoint dashboard. By using this enabler, it is possible to assemble all or only some of the components in the enabler architecture. This brings flexibility to the continuous monitoring since it is possible to tailor the enabler design to the particular needs of the SIS under study in terms of e.g., how many security agents are deployed and where, which security policies are used by the clients, which metrics are monitored, and the needed tailored alarms and data visualisations can be created ad hoc.

The continuous monitoring offered is holistic in the sense that it correlates data captured in the three main layers of the SIS: network, system and application layers. And this makes possible a high richness and accuracy of security incidents and anomalies detection which leverages signature-based detection together with machine learning and deep learning-based detection.

The enabler also answers to the needs of rapid elasticity and full scalability required by SIS that involve large amounts of sensors and actuators, while it still is able to offer the required visualisations and notifications that constitute the basis for the informed situational awareness of the overall system.

In order to be able to take advantage of the insights gained by the tool over the SIS, the enabler was designed with a security event bus for integration with other cybersecurity threat intelligence platforms and services, such as those of forensics analysis and cybersecurity information sharing with third parties.

Last but not least, the enabler design permits a seamless integration with controls at application layer, for example those developed on top of the SOFIA SMOOL IoT platform monitoring and control agents' management, which are able to monitor and control secure communications among the smart things of the SIS.

As part of the future lines of work in the enabler, the automatic reaction capabilities will be researched and enriched by extending the controls with intelligent security orchestrators and decision making support that facilitate the combination of multiple reaction measures when needed in the different layers of the SIS, so as the security level of the system is increased in the face of attack or incident symptoms.

References

[1] Rego *et al. SMOOL source code.* https://bitbucket.org/jasonjxm/smool, 2011–2020.

[2] Elasticsearch B.V. *The Elastic Stack. Retrieved March 11, 2021*, 2021. URL: https://www.elastic.co/elastic-stack.

[3] Prosunjit Biswas, Ravi Sandhu, and Ram Krishnan. *Attribute Transformation for Attribute-Based Access Control.* https://profsandhu.com/confrnc/misconf/abac17-prosun.pdf, 2017.

[4] Apache Software Foundation. *Apache Kafka. Retrieved March 11, 2021*, 2017. URL: https://kafka.apache.org.

[5] Dick Hardt. *The OAuth 2.0 Authorization Framework.* 2012. URL: https://tools.ietf.org/html/rfc6749.

[6] Wazir Zada Khan *et al.* "Industrial internet of things: Recent advances, enabling technologies and open challenges". In: *Computers & Electrical Engineering* 81 (2020), p. 106522.

[7] NIST. *NIST Computer Security Resource Center Glossary. National Institute of Standards and Technology. Retrieved March 11, 2021*. 2020. URL: URL=%20 https://csrc.nist.gov/glossary/term/information_security_continuous_monitoring.

[8] NIST. *NIST SP 800-53 Rev. 5. (December 2020). Security and Privacy Controls for Information Systems and Organizations. National Institute of Standards and Technology. Retrieved March 11, 2021*. 2020. URL: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final.

[9] Adrian Noguero, Angel Rego, and Stefan Schuster. "Towards a Smart Applications Development Framework". *Social Media and Publicity* 27 (2014). URL: https://bitbucket.org/jasonjxm/smool,%202011-2020.

[10] OASIS. *eXtensible Access Control Markup Language (XACML) Version 3.0. Retrieved July, 2019*. 2019. URL: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

[11] OpenID. *OpenID Connect*. 2014. URL: https://openid.net/connect/.

[12] Inger Anne Tøndel, Maria B. Line, and Martin Gilje Jaatun. "Information security incident management: Current practice as reported in the literature". In: *Computers & Security* 45 (2014), pp. 42–57.

[13] The Board of Trustees of the Leland Stanford Junior University. *Protégé, open-source ontology editor and framework*. 2020. URL: https://protege.stanford.edu/.

DOI: 10.1561/9781680838251.ch8

Chapter 8

Validation, Verification and Root-Cause Analysis

By Luong Nguyen, Vinh Hoa La, Wissam Mallouli and Edgardo Montes de Oca

8.1 Motivation

Verification and validation are two significant software development processes for checking that software meets its requirements and specifications and fulfills its intended purpose. In these processes, various test cases (e.g., unit tests, integration tests, regression tests, system tests) need to be designed and executed in a production-like environment that reproduces the same conditions where the software under test would run. However, having access to such an environment is usually tricky or close to being an impossible task. It is even particularly challenging in the IoT arena. The access to IoT devices might be nontrivial or limited due to many factors. Networks of physically deployed devices are typically devoted to production software. Testing applications on top of those networks might involve additional testing software, which might affect overall performance and the revenue generated by the devices (e.g., applications need to be stopped to load their new versions).

Software simulators proved to be valuable in easing the verification of the software requirements. They provide software developers a testing environment to at least manage the execution of test cases. IoT Testbeds play a similar role in testing IoT applications. They offer a deployed network of IoT devices where developers can upload their applications and test their software in a physical environment. IoT-Lab [1] and SmartSantander [9] are good examples of IoT testbeds. Testbeds often have a predefined fixed-configuration and architecture. They are also usually shared with other users, which can be a problem for measuring application quality. Hence, this problem might make simulators more attractive since they provide a more customized and controlled environment. Furthermore, simulators avoid the need for a more expensive physical network of devices.

In recent years, both academia and the commercial market have proposed solutions for the IoT simulation field. These solutions are often entirely different, although their objectives are similar. The academic solutions implement cutting-edge technology as proofs-of-concept, and they are usually not ready for production systems. By contrast, the commercial solutions are designed to be stable and flawless, even though the technology behind them might not be state-of-the-art.

The ENACT project has brought an opportunity to create the Test and Simulation (TaS) tool. Collaborating with universities and research institutions such as SINTEF and CNRS, we provide a state-of-the-art test and simulation tool with cutting-edge technology behind it. We have evaluated our solution with several industrial use cases, such as eHealth (Tellu), Smart Building (Tecnalia), and Intelligent Train System (Indra). The case studies have shown that it is stable and ready for production systems.

The TaS provides the possibility to test the IoT system based on test scenarios using pre-prepared datasets. The datasets can be the recorded data from a real system or the data generated using some data mutation operators. The TaS also allows stressing the boundaries of the scenarios to detect potential problems.

We focus on the network of sensors and the applications on top of them. Therefore, we do not consider the physical behavior of the sensors. We take it for granted that the sensors are reliable and correctly react to the physical changes (e.g., if the physical temperature rises 2 degrees, the sensor will immediately send a message with a 2 degree higher reading).

On the other hand, it is also important to note that failures usually propagate in complex systems through causal chains and produce evolving fingerprints of noisy symptoms. One of the first tasks to accomplish for an automated tool helping humans troubleshoot a system is to group events that are causally connected (and keep unrelated events separated). Achieving this is often not straightforward since components of a system can exhibit similar symptoms of two unrelated failures. We need a higher level of granularity in the monitoring indicators and a deeper analysis to distinguish two unrelated failures. Moreover, it is frequent that failures are recurrent. The system administrators, who have some experience dealing with failures, can react more quickly and efficiently against their recurrence. They can take the impact estimation and the mitigation action (e.g., reset a particular server every night) promptly.

Indeed, all aforementioned points lead to the need for a Root-Cause Analysis (RCA) tool which enables systematizing the experience in dealing with faults and problems to identify the root cause of a newly detected issue. Thanks to RCA results, remediation actions and reactions could be timely and wisely taken to prevent or mitigate the damage of the recurrence of problems.

The IoT world has promised to connect everything and create systems with an enormous number of devices. The need for RCA to implement and operate IoT systems is evident; IoT represents a generic framework that an RCA solution can target. However, several characteristics of these types of systems need to be considered: First, IoT networks are often very dynamic environments, with devices frequently joining and leaving a system (e.g., mobile devices connecting to a particular antenna). Nevertheless, most of the communications are likely to be wireless. This can introduce a higher degree of unreliability. The failure, however, can present symptoms very similar to a normal activity. For example, when we no longer receive sensed data from a sensor, it is difficult to determine whether the sensor is no longer in range or the communication has failed. Second, in many cases, the number of components/ indicators to be taken into the analysis could be enormous. This can lead to a big volume of data processed. Reducing the data dimension by avoiding less relevant attributes (i.e., noises) is a natural need. Finally, battery-powered devices may have a low-activity mode to extend their operation autonomy. In this mode, inputs may not be synchronized and have the same frequency as other information RCA uses for the diagnosis. Therefore, RCA must be able to deal with out-of-order data. In the context of ENACT, our RCA enabler would try to address all the challenges we mentioned above.

In summary, this chapter focuses on TaS and RCA, two primary parts empowering the validation and verification in an IoT DevOps cycle, which have been developed and evaluated in the context of the ENACT project. To the best of our knowledge, no similar tool had ever been created for IoT. On the one hand, the TaS tool enables the simulation and testing of an IoT system. It collects the events of a running IoT system without impacting its normal behavior. The recorded events can be used to simulate the system, inject different kinds of "problems", and collect all relevant data for detecting errors, failures, and unwanted symptoms. On the other hand, the RCA tool monitors the real system and performs the diagnostic analysis when some errors or failures occur. The enabler allows determining unknown incidents' symptoms and evaluating how much the unknown incident is similar to a known/learned one. We discuss more technical details regarding TaS and RCA in the following sections.

8.2 Test and Simulation (TaS)

In this section, we first present an overview of the TaS enabler. We then give the details of the enabler.

8.2.1 Overview and Approach

The TaS enabler is a test and simulation solution well adapted to IoT environments. It allows simulating different IoT topologies and performing various tests to detect potential errors and security failures. We first present the main features and components of the enabler that simulate an IoT system. Then, we give a detailed description of the TaS enabler architecture.

8.2.1.1 Smart IoT system components

Figure 8.1 shows some main components in a Smart IoT System:

- **The Sensor Node:** captures, pre-process, and sends the sensor data to the gateway that can be a Raspberry PI, an Arduino, etc. It implements some basic modules:
  - The Sensor module captures the environment information.
  - The Onboard Processing module reads the sensor data and pre-processes it (e.g., performs calculations and formalizes and validates data). It can be an IoT application, a Node-RED flow, etc.
  - The Communication module component communicates with the gateway to send or broadcast the processed data.
- **The Gateway device:** receives data from the sensor nodes and processes or just forwards it to the other components/services such as cloud-based application and control center.
- **The Actuator node:** reacts and controls the actuator based on the reactions of the IoT system. It contains some basic modules:
  - The Actuator module triggers a change on the IoT device, such as opening a door and activating an alarm system, etc.
  - The Onboard processing module reads the actuator data signal and converts it into an action.
  - The Communication module communicates with the gateway to receive the actuation data signal.
- **Other components:** Other components are higher-level components that can provide a service or an application that receives and processes the data and performs actions depending on the business logic.

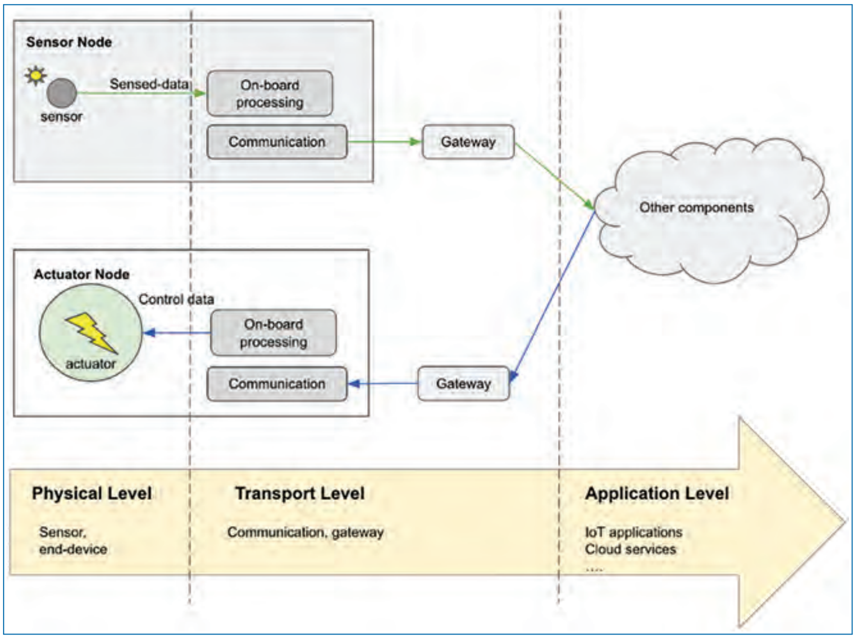


Figure 8.1. IoT system components.

The SIS components can be organized in a multi-layer architecture which we present in the next section.

8.2.1.2 Simulating a smart information system (SIS)

An SIS can be represented in three levels depicted in Figure 8.1. The physical level contains all the physical components, such as sensors and actuators, produced by a manufacturer and cannot be changed by developers. The transport level is responsible for transmitting the data within the SIS network. Developers can configure the transport level to use a specific port number or protocol. Finally, the highest level is the application level, which contains the application written by developers. The application receives the data from sensors, processes it, and produces an action to be performed by the actuators (e.g., turn the light on or off). Software engineers usually work on the application level.

When it comes to developing a software application, a software application needs to be tested every time there is a change in its source code or in the infrastructure it uses. The planned tests aim to cover many scopes involving different testing scenarios. While coping with multiple test scenarios, the testing environment needs to be flexible for manipulating input and measuring output. It is not easy to have such a testing environment for IoT applications since sensors at the physical level depend on the physical environment. Therefore, only the scenarios matching the current condition of the environment can take place.

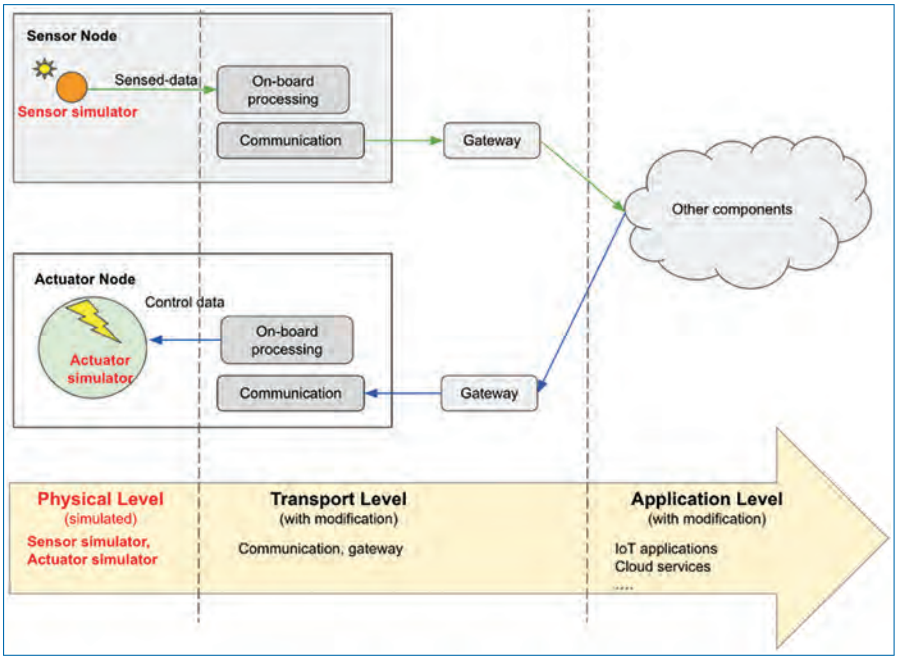


Figure 8.2. An IoT network architecture with simulated components.

Within an IoT network, a sensor captures the information of its surrounding environment at a specific time. This information is transformed into a digital format. Since it is not reasonable to wait for the change in the environment to test IoT applications, simulating various sensor measurements is very beneficial for testing them. It allows the developer to control sensor values and, thus, to simulate and test the IoT application in all scenarios without waiting for environment changes.

An actuator presents the SIS reactions in a specific situation, for example, switching on a light bulb. In such cases of testing system reactions, it is sufficient to measure the actuated data sent by the SIS to actuators. In the TaS enabler, the actuator is simulated by simply creating a hub to receive the actuated data instead of using a real actuator. Note that the impact of an actuator on a sensor is not yet considered.

With the TaS enabler, we only need to simulate the components at the physical level. The other (software) components can be cloned from the system under test and configured to work in a classical test and simulation environment, avoiding the need for communicating with a production environment, as we can see in Figure 8.2.

8.2.1.3 The TaS enabler's global approach and architecture

In this subsection, we present the architecture of the TaS enabler, which is based on the concept of Digital Twins [3]. Figure 8.3 illustrates the TaS enabler architecture.

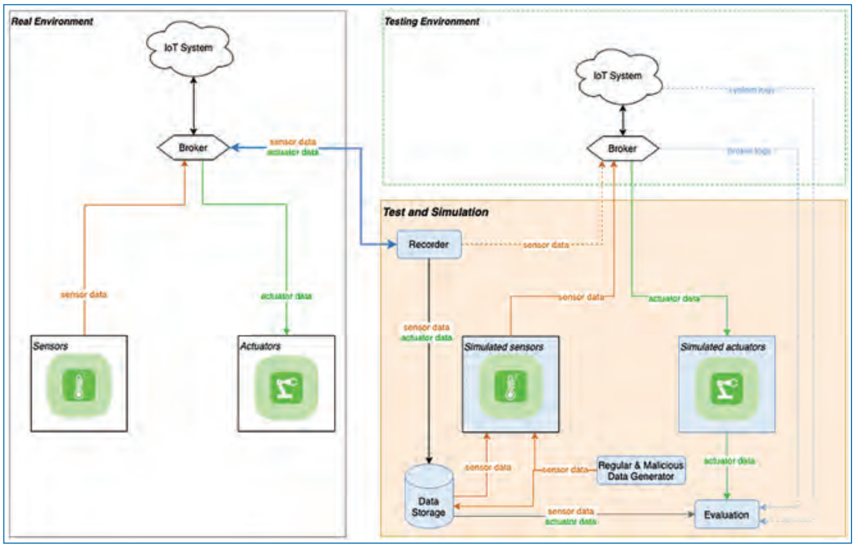


Figure 8.3. Test and Simulation (TaS) Enabler approach and architecture.

On the left-hand side, we have the system in a real (production) environment. The communication between the sensors, actuators with the IoT component is typically done via a broker. The sensors capture and send the surrounding information (e.g., temperature) to the IoT system. Based on input data, the IoT system reacts differently and sends actuation data to change the actuator settings (e.g., "change the heating level").

On the right-hand side of the figure, we have the SIS in a test environment and the TaS enabler. The system under test is the SIS that needs to be tested. The TaS enabler simulates sensors and actuators. The topology on the left side is very similar to the topology on the right side. The only difference is the simulated sensors and actuators. The simulated actuators collect the actuation data sent from the IoT system. The simulated sensors play the same role as the physical sensors providing the data signal to the IoT components. However, they are much more valuable than a physical sensor in terms of testing in the following ways:

- Firstly, by using the dataset recorded from the physical environment, the simulated sensors can repeatedly simulate the surrounding environment at a specific time. In reality, an event may happen only once, but the simulated sensor can generate the same event as many times as needed for testing purposes.
- Secondly, the physical sensors passively capture the state of the surrounding environment. It can be challenging to obtain different data from the physical sensors. In contrast, the simulated sensors use the dataset in the Data Storage as a data source. Therefore, we can generate various testing scenarios by modifying the event in the Data Storage.

- Moreover, the TaS enabler also provides a module to manipulate the data from the sensors. The Regular and Malicious Data Generator can generate regular data to test the functionalities, operations, performance, and scalability. It can also generate malicious data to test the resiliency of the system to attacks.

Besides the simulated sensors and actuators, the TaS enabler also provides some modules which support the testing process 8.3. The *Data Recorder* module records all the messages going through the broker in the physical environment. Each message can be considered as an event happening in the physical environment. Then, the recorded messages are forwarded to the broker in the testing environment. In this way, we have a "twin version" of the physical environment. What has happened in the physical environment is reproduced in the testing environment. Besides, the recorded messages are stored in a *Data Storage* as a dataset for later testing. The recorded dataset can be modified (muted) to create a new dataset, e.g., "change the event order", "delete an event", "add a new event". All the testing datasets are stored in the Data Storage. The *Regular and Malicious Data Generator* enables the simulation of different sensor behaviors, from normal behavior to abnormal behavior, such as a DOS attack (the sensor publishes massive data messages in a short time), node failure (the sensor stops sending data). With data mutation, the TaS enabler can help build datasets for testing many different cases hard to produce in real life. Finally, the *Evaluation* module analyses the simulation input and output and combines them with the logs collected from the IoT system to provide the final result of a testing process.

The next section presents more details on how the TaS enabler simulates an SIS.

8.2.2 Simulation of a Smart IoT System

Most of the testing scenarios are defined by the information about the surrounding environment captured by sensors. The following section goes into detail about the simulation of sensors.

8.2.2.1 The simulation of sensor

The sensor provides the input data of an IoT system. The simulation of a sensor corresponds to the simulation of the data stream it provides. The simulated sensor has been designed for flexibility in the following ways:

- It supports different types of data report formats:
  - PLAIN_DATA: the measurement value is published directly without any transformation, it can be a number, a string or an object, for example: 15

- JSON_OBJECT: the measurement value is transformed to be an object in JSON format, with the key is set by user, for example: "temp":15
- IPSO_FORMAT: the reported data follows the Internet Protocol for Smart Object (Object and Resource Registry). A temperature sensor can report the data in IPSO format as follows (Temperature Sensor in IPSO):

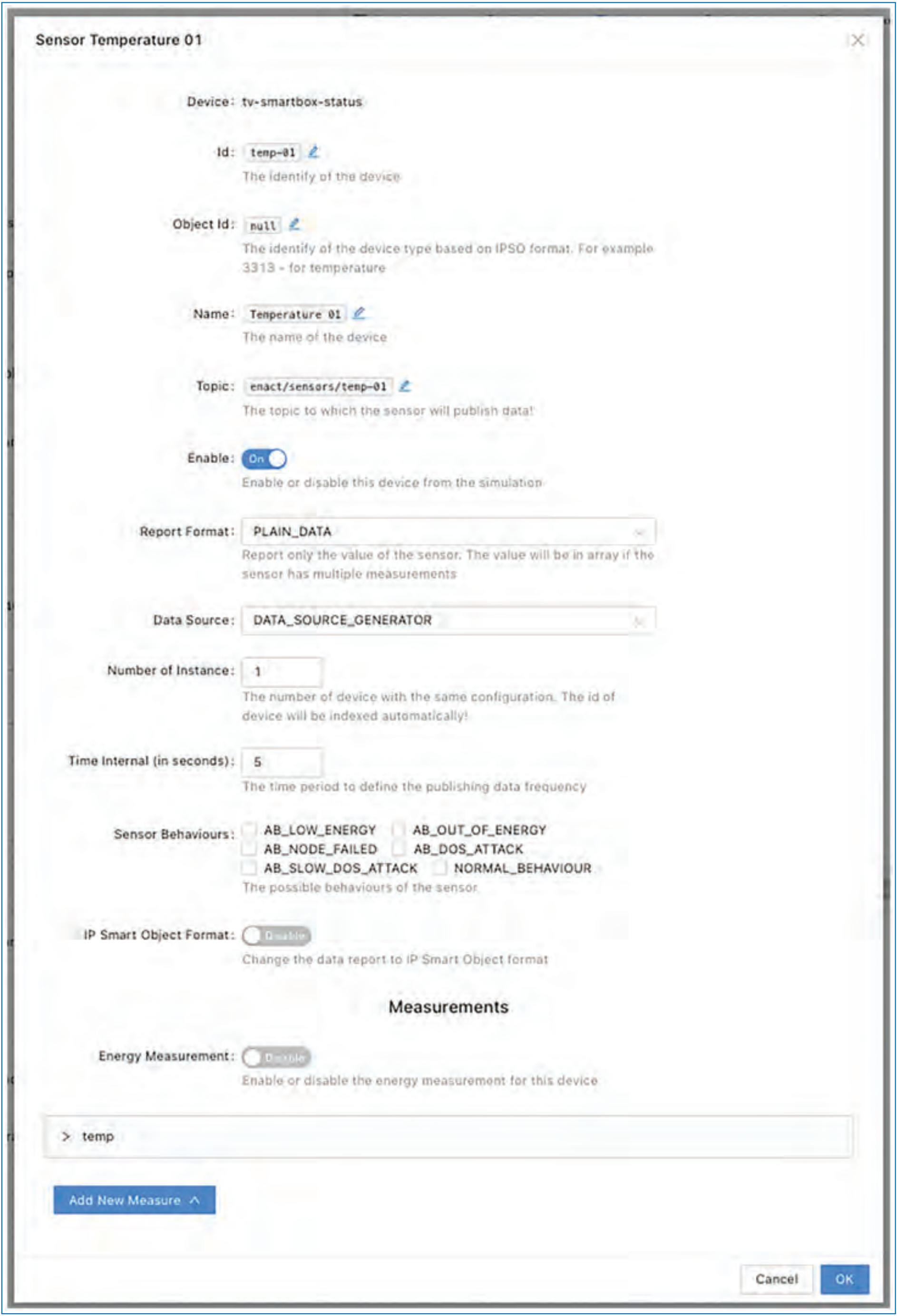
```
{
    "InstanceId": 5,
    "ObjectId": 3303,
    "TimeStamp": 1601498832,
    "TimeAccuracy": 364449977,
    "Resources": {
        "5700": 15,
        "5701": "celcius"
    }
}
```

- It supports different data sources which are used for simulation:
  - Dataset: The data source is from the data storage where the data has been recorded or created before simulating.
  - Data Generator: The data will be generated at run-time during the simulation.
  - Data Recorder: The data source is the data recorded from a real system and forwarded to the testing system.
- It supports simulating several abnormal behaviours, such as, low energy, node failure, DOS attack, and slow DOS attack.
- It supports multiple measurements with the different data types, such as Boolean, Integer, Float and Enum. For each measurement, there are several abnormal behaviours that can be selected, such as “fixed value”, “value out of range”, and “invalid value”.

Figure 8.4 presents the definition of a temperature sensor, which generates (data source: *DATA_SOURCE_GENERATOR*) a measurement value every 5 seconds. The measurement value is published in *PLAIN_DATA* format to a MQTT/MQTTS message bus communication channel defined by the topic *enact/sensors/temp-01*. The sensor does not have any abnormal behaviour.

8.2.2.2 The simulation of actuator

An actuator can be considered as a device that receives the IoT system reaction based on the input data. We simulate the actuator as a component that will receive the reaction signal (actuation data) from the IoT system. Figure 8.5 shows the configuration of a Heater. The actuator listens for the actuation data on an MQTT/MQTTS message bus communication channel defined by the topic:



Sensor Temperature 01

Device: tv-smartbox-status

Id: temp-01
The identify of the device

Object Id: null
The identify of the device type based on IPSO format. For example 3313 - for temperature

Name: Temperature 01
The name of the device

Topic: enact/sensors/temp-01
The topic to which the sensor will publish data!

Enable: On
Enable or disable this device from the simulation

Report Format: PLAIN_DATA
Report only the value of the sensor. The value will be in array if the sensor has multiple measurements

Data Source: DATA_SOURCE_GENERATOR

Number of Instance: 1
The number of device with the same configuration. The id of device will be indexed automatically!

Time Internal (in seconds): 5
The time period to define the publishing data frequency

Sensor Behaviours: AB_LOW_ENERGY, AB_OUT_OF_ENERGY, AB_NODE_FAILED, AB_DOS_ATTACK, AB_SLOW_DOS_ATTACK, NORMAL_BEHAVIOUR
The possible behaviours of the sensor

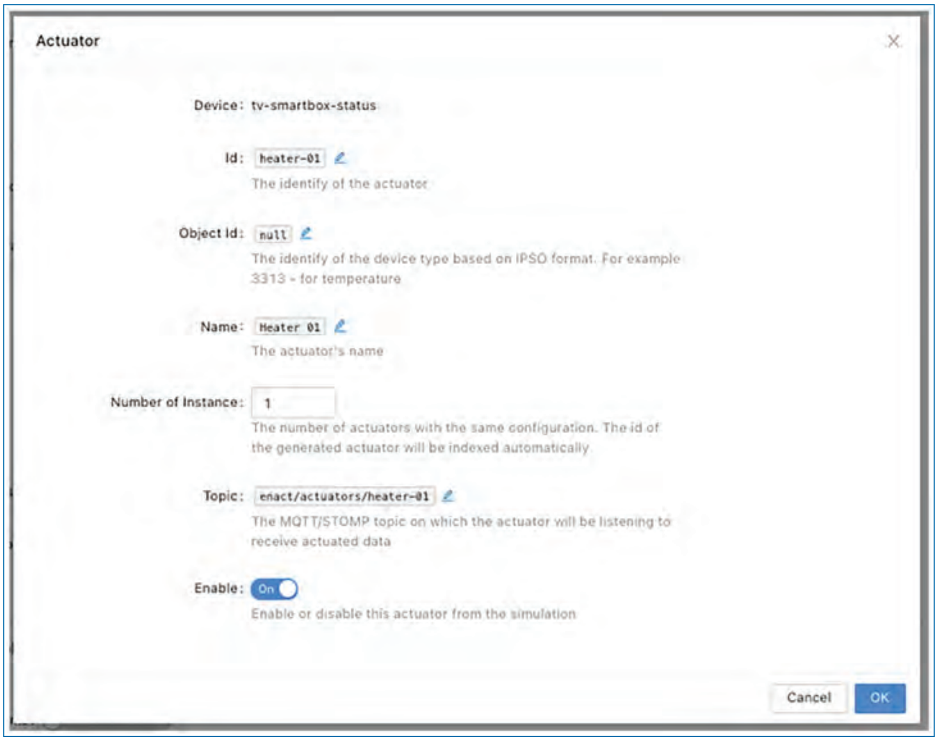
IP Smart Object Format: Disable
Change the data report to IP Smart Object format

Measurements

Energy Measurement: Disable
Enable or disable the energy measurement for this device

> temp

Figure 8.4. A temperature sensor.


Actuator

Device: tv-smartbox-status

Id: heater-01
The identify of the actuator

Object Id: null
The identify of the device type based on IPSO format. For example 3313 - for temperature

Name: Heater 01
The actuator's name

Number of Instance: 1
The number of actuators with the same configuration. The id of the generated actuator will be indexed automatically

Topic: enact/actuators/heater-01
The MQTT/STOMP topic on which the actuator will be listening to receive actuated data

Enable: On
Enable or disable this actuator from the simulation

Figure 8.5. A Heater actuator.

enact/actuators/heater-01. The topic defines the channel on which the actuator will connect to obtain the actuation data.

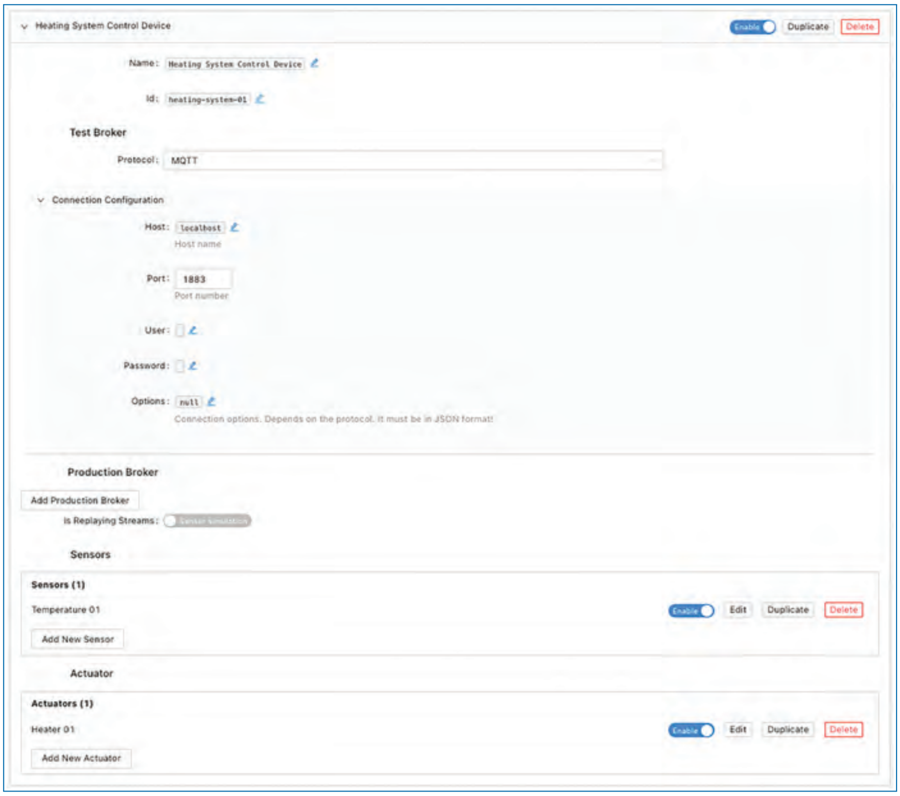
8.2.2.3 The simulation of an IoT device

In an IoT system, the sensor and actuator are usually part of the same device. An IoT device can contain one to many sensors as well as one to many actuators. Figure 8.6 illustrates the configuration of a Heating System Control device. The device has one sensor and one actuator. The data is published by the sensor and received by the actuator via the MQTT protocol.

8.2.2.4 The simulation of a network topology

Figure 8.7 presents a simple simulated network topology.

A list of simulated IoT devices forms the simulated network topology. Besides the list of devices, a network topology can also provide the identifier of the dataset (*datasetId*),which contains the data to simulate the SIS in a given time, the global replaying options, the configuration to connect with the database, and


Heating System Control Device

Name: Heating System Control Device

Id: heating-system-01

Test Broker

Protocol: MQTT

Connection Configuration

Host: localhost
Host name

Port: 1883
Port number

User:

Password:

Options: null
Connection options. Depends on the protocol. It must be in JSON format!

Production Broker

Is Replaying Streams:

Sensors

Sensors (1)

Temperature 01

Actuator

Actuators (1)

Heater 01

Figure 8.6. A Heating System Control device.

the definition of the new dataset where the data generated from the simulations will be stored.

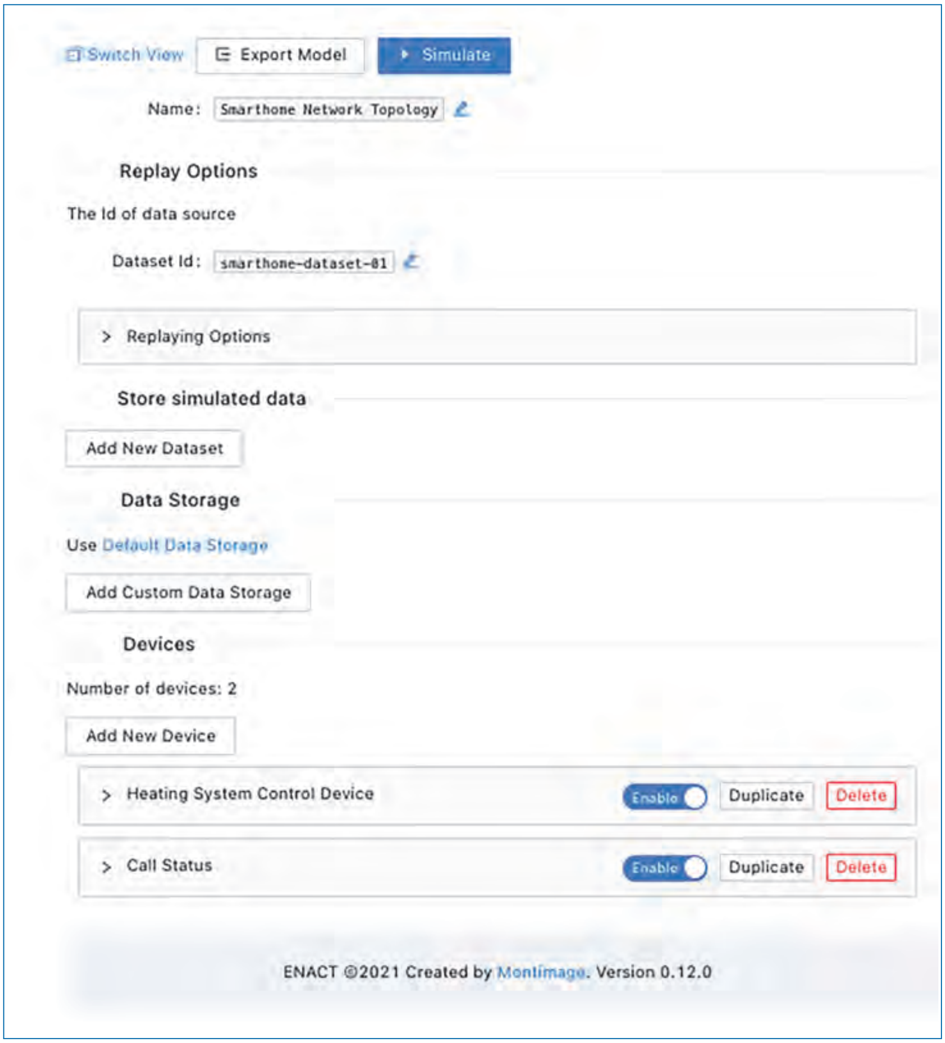
8.2.2.5 The communication between the TaS enabler and the system under test

In the ENACT project, the communication between the TaS enabler and the system under test has been implemented based on message queue protocols such as MQTT and MQTTS. Figure 8.8 presents the Message Bus class diagram, similar to the interface for all Message Queue Protocols.

Some basic message queue bus protocol methods have been implemented, such as subscribe, unsubscribe, publish, connect, and close. By design, each IoT device can have its way of communication with the SIS systems.

8.2.3 The Testing of a SIS

In this section, we present the testing methodologies and techniques we have adapted in the TaS enabler.



Switch View | Export Model | Simulate

Name: Smarthone Network Topology

Replay Options

The Id of data source

Dataset Id: smarthone-dataset-01

> Replaying Options

Store simulated data

Add New Dataset

Data Storage

Use Default Data Storage

Add Custom Data Storage

Devices

Number of devices: 2

Add New Device

> Heating System Control Device | Enable | Duplicate | Delete

> Call Status | Enable | Duplicate | Delete

ENACT ©2021 Created by Montimage. Version 0.12.0

Figure 8.7. Smarthome network topology.

8.2.3.1 The testing methodologies

This section covers the testing methodologies that the TaS enabler can support. In this first version of the enabler, we have implemented only data-driven and data-mutation testing methodologies. The other ones described below are possible future extensions.

Data Driven Testing

Figure 8.9 presents the data flow of the Data-Driven Testing method. The Data Storage contains the datasets recorded from the IoT system or entered manually. Each dataset contains sensor data (inputs for TaS) and expected actuator outputs.

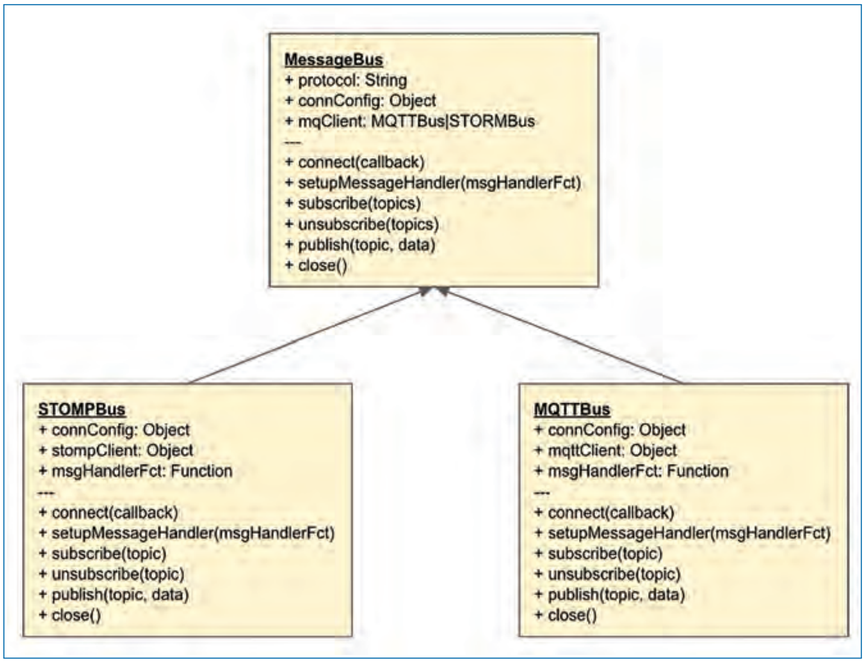


Figure 8.8. Message queue bus class diagram.

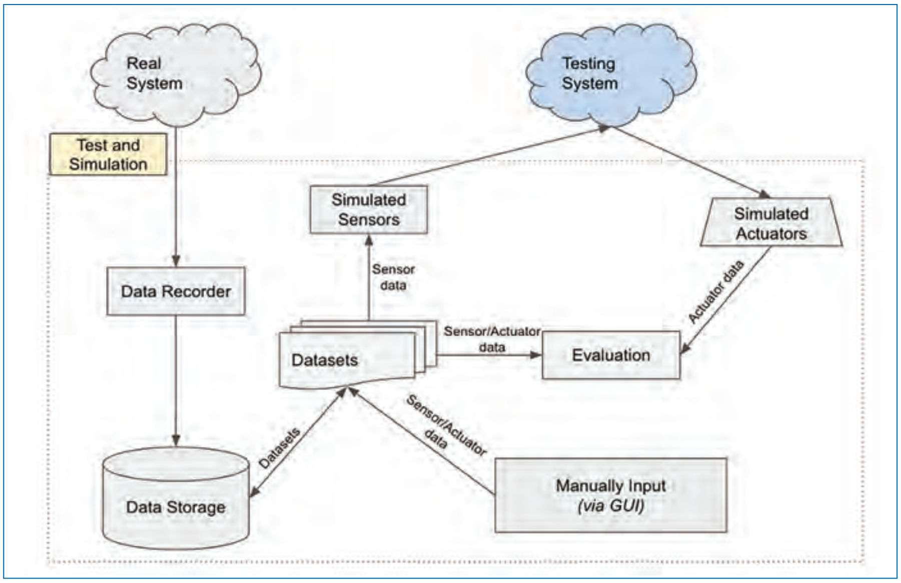


Figure 8.9. Data Driven Testing.

The expected actuator outputs can be the value recorded from the IoT system in a normal scenario. Engineers can also enter them manually via the Graphical Interface. The Evaluation module will use the expected outputs to compare them with the simulation output to determine if they match. A test case passes if the simulation output is the same as the expected output. The Data-Driven Testing method is suitable for functional and regression testing.

The Data-Driven Testing has been implemented as the main testing methodology of the TaS enabler.

Data Mutation Testing

Figure 8.10 illustrates the Data Mutation Testing architecture. The Mutant Generator generates new sensor data from existing data stored in the Data Storage by applying one or many mutated functions, such as "change the event order", "change a value", and "delete an event". The mutated data are input for the simulation. The Evaluation module generates a report about the output differences when testing the system with the mutated and the original input data. The Data Mutation Testing method is for penetration, robustness, security, and scalability testing (e.g., mutating the device identifier to obtain new devices). In the TaS enabler, we can mutate the device identity to generate many devices while testing the system scalability. There is also an interface to apply some mutation functions to a dataset manually.

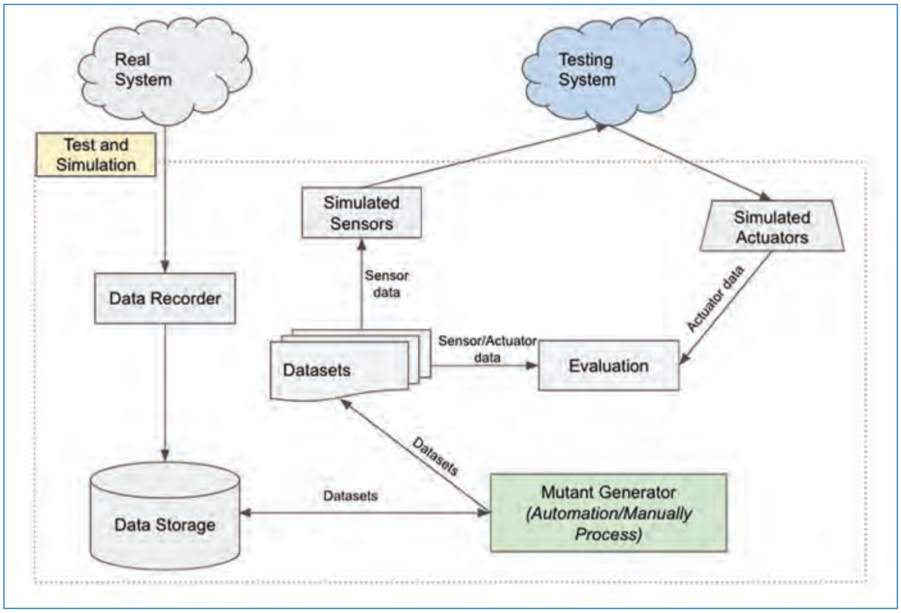


Figure 8.10. Data Mutation Testing.

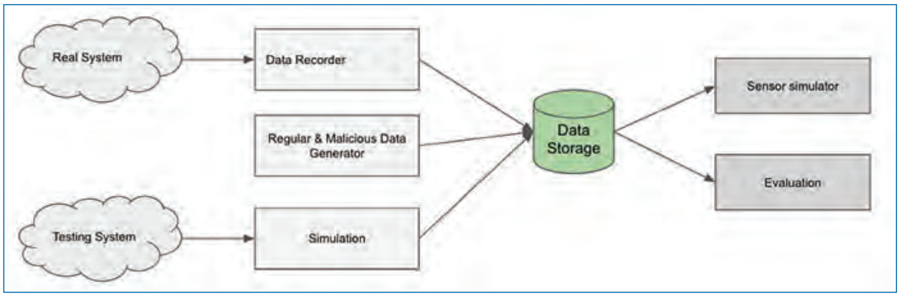


Figure 8.11. DataStorage.

Model-Based Testing [10] and Risk-Based Testing [6] are two other methodologies that we have studied but not yet implemented in the TaS enabler at the time of writing of this book chapter.

8.2.3.2 The testbeds

To simulate and test an IoT system in some specific scenarios, one of the easiest methods is to use a testbed. With testbeds, the developer can define exactly what is the input and what should be the corresponding output. This way, the tests can be done automatically and easily integrated in the DevOps Continuous Integration and Continuous Deployment processes. In TaS, testbeds are built from datasets which are recorded from a real system or generated by the TaS based on scenarios.

DataStorage

The Data Storage contains all the datasets for testing and simulating.

As depicted in Figure 8.11, the datasets are fed into the DataStorage via three sources: the data from the real system recorded by the Data Recorder, the data generated by the Regular and Malicious Data Generator, and the data generated by the simulation. The datasets in the Data Storage are used to simulate the sensors and to validate the simulation output.

The database connection of the TaS enabler is flexible. Two simulations can use different databases. If there is no configuration specified, the TaS enabler uses a default database. The database to connect to can be any database that the TaS enabler can reach.

Event

An event represents a message sent through the communication channels. It can be a data message sent by a sensor or data received by an actuator. Figure 8.12 presents the format of the event Schema.

The *timestamp* attribute indicates the time when the event has been captured. The *topic* represents the MQTT/MQTTS bus channel related to the event. It is the

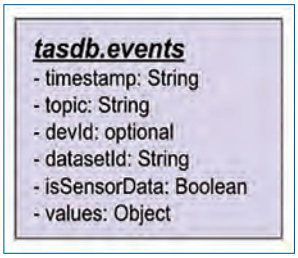


Figure 8.12. Event Schema.

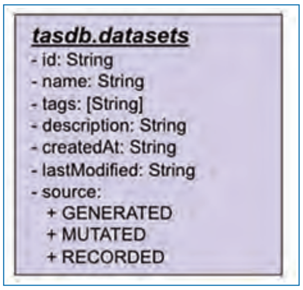


Figure 8.13. Dataset Schema.

source channel of the event where the data message corresponds to sensor data. It is the destination channel of the event where the data message is data received by an actuator. Its value is crucial for identifying the event to be replayed. The *datasetId* attribute represents the dataset to which the event belongs. The *isSensorData* is set to True if the event presents a data message sent by a sensor. It is False if the event is a data message received by an actuator. The *values* attribute contains the value of the message data. This value can be a number, a string, or an object. This design helps make the event generic and making it possible to consider any message data type.

Dataset

A dataset contains a series of events for a specific scenario. Figure 8.13 presents the schema of a dataset. Each dataset has a unique *id*, *name*, and *description* to describe the dataset objective. The *source* attribute indicates the dataset source. A dataset can be created from a recording session by the Data Recorder (source: RECORDED) or generated by the Regular and Malicious Data Generator (source: GENERATED). A dataset can also be derived by cloning and modifying data from another dataset (source: MUTATED).

By grouping the events by the dataset Id, we have all the events belonging to a dataset.

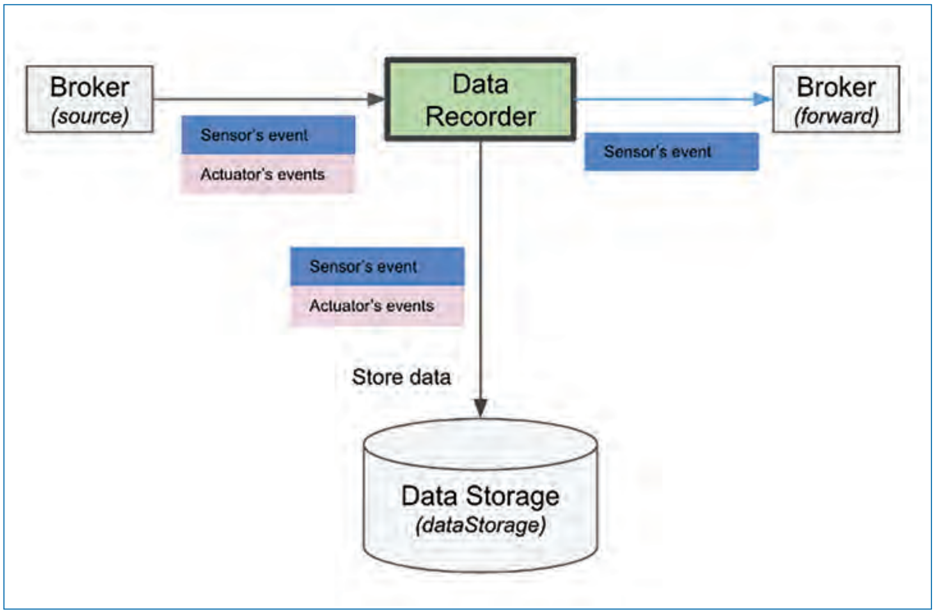


Figure 8.14. Data Recorder data flow.

8.2.3.3 The data recorder and digital twins concept

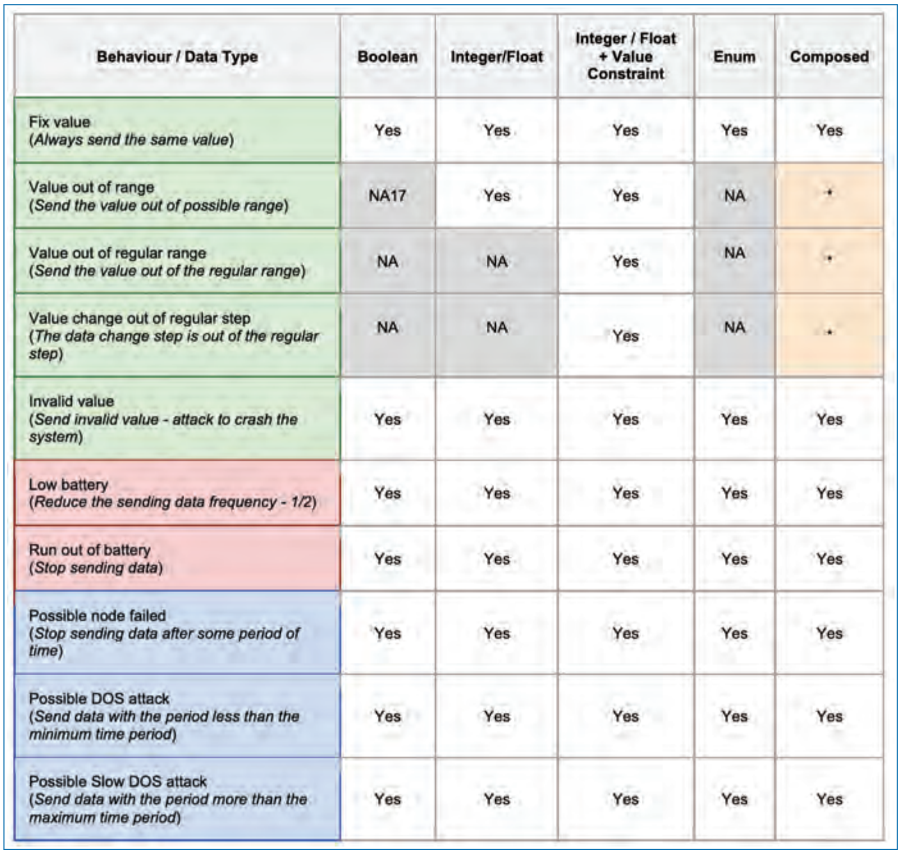
The TaS enabler provides the possibility to simulate an IoT system using historical data. To this end, a Data Recorder module is needed.

Figure 8.14 presents the data flow of the Data Recorder. All the events in the real system (coming from the broker) will be recorded. This data (including both sensor and actuator data) is stored in the Data Storage as a dataset. The sensor data can be forwarded directly to the testing system (using the forwarding broker). With the recorded data from the real system, the SIS can be tested with real input. The more data from sensors are recorded, the more test scenarios are tested. By synchronizing the Sensor simulator timestamps with the Data Recorder, it is possible to simulate a particular SIS (following the Digital Twin concept). By monitoring the SIS input and output, we can build an automatic testing process for a complex IoT system.

The recorded data can be used as a source for simulation. It can also be mutated so that it can contain different values for obtaining a modified testing scenario. In the next section, we will explain how to generate a new dataset using a given behavior profile.

8.2.3.4 The regular and malicious data generator

When testing the IoT system, there are many testing scenarios and cases that do not frequently occur in reality. With the real IoT system, it is almost impossible to collect the datasets for many testing scenarios. The TaS enabler provides a powerful



Behaviour / Data Type	Boolean	Integer/Float	Integer / Float + Value Constraint	Enum	Composed
Fix value *(Always send the same value)*	Yes	Yes	Yes	Yes	Yes
Value out of range *(Send the value out of possible range)*	NA17	Yes	Yes	NA	*
Value out of regular range *(Send the value out of the regular range)*	NA	NA	Yes	NA	*
Value change out of regular step *(The data change step is out of the regular step)*	NA	NA	Yes	NA	*
Invalid value *(Send invalid value - attack to crash the system)*	Yes	Yes	Yes	Yes	Yes
Low battery *(Reduce the sending data frequency - 1/2)*	Yes	Yes	Yes	Yes	Yes
Run out of battery *(Stop sending data)*	Yes	Yes	Yes	Yes	Yes
Possible node failed *(Stop sending data after some period of time)*	Yes	Yes	Yes	Yes	Yes
Possible DOS attack *(Send data with the period less than the minimum time period)*	Yes	Yes	Yes	Yes	Yes
Possible Slow DOS attack *(Send data with the period more than the maximum time period)*	Yes	Yes	Yes	Yes	Yes

Figure 8.15. Abnormal behaviours based on the data type and the constraints.

tool to solve this problem. The Regular and Malicious Data Generator module helps developers create a testbed. It enables generating sensor data for various scenarios, e.g., making the temperature too high or too low. By combining multiple data, one can create a testbed that includes many incidents or attack scenarios, such as DDoS and data poisoning. The Data Storage stores all the generated data for further use. Based on the data type and constraints on the time, values, or energy use, many abnormal behavior types may exist as depicted in Figure 8.15.

The abnormal sensor behaviors are defined by energy, reporting time, and value constraints.

Figure 8.16 illustrates how a data value is generated based on the selected behaviours of the sensor. In the beginning, the energy constraint is checked. There are two behaviors related to energy. If the sensor is in the low-battery mode, we can reduce the reporting frequency. Notice that the user initially sets the frequency. If the sensor is out of battery, it stops sending data. In the next step, we consider the time constraint. We can select among three behaviors. The possible DOS attack

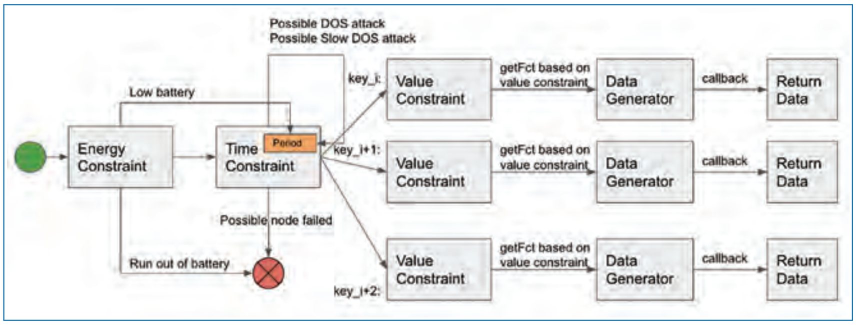


Figure 8.16. Data generating flow.

increases the reporting frequency to simulate a DOS attack, and the sensor sends a lot more data than what is considered normal. If there is a constraint on the maximum delay time for reporting data, we can simulate a behavior like a possible slow DOS attack. For example, if the system expects to receive the temperature information every 5 seconds maximum, then a slow DOS attack could change the sensor behavior that the sensor sends a message every 6 seconds. Based on the time constraint, we can simulate a sensor that stops sending data for a certain period (node failed). Finally, for each measurement provided by a sensor, the value constraint is checked. There are many behavior types based on the measured data type, such as invalid value or fixed value. Based on the selected behavior, the Data Generator function returns a specific value for the measurement.

Besides the sensor's behavior, it is also possible to change the behavior of the IoT devices. For example, the GATEWAY_DOWN behavior makes the simulated IoT device stop working after some time. When an IoT device stops working, all the sensors and actuators belonging to that device also stop working.

8.2.3.5 Automatic testing

The TaS enabler has been designed to be easily integrated into any Continuous Integration and Continuous Delivery processes. Figure 8.17 illustrates the TaS enabler concept. One of three events trigger the TaS process: code commit, new component (software module or hardware device) added, new scenario added. Several tests are executed by simulating the different testing scenarios on the system under test. The tests can cover functional, operational, security, performance, and scalability testing. If all the tests pass, we can deploy the new changes in the real environment.

Following the process we depict above, we can automatically test every change in the system and cover every test scenario.

Test case

While testing an IoT system, we may want to test different network topologies, such as adding a new device, removing a device, or just changing the way to

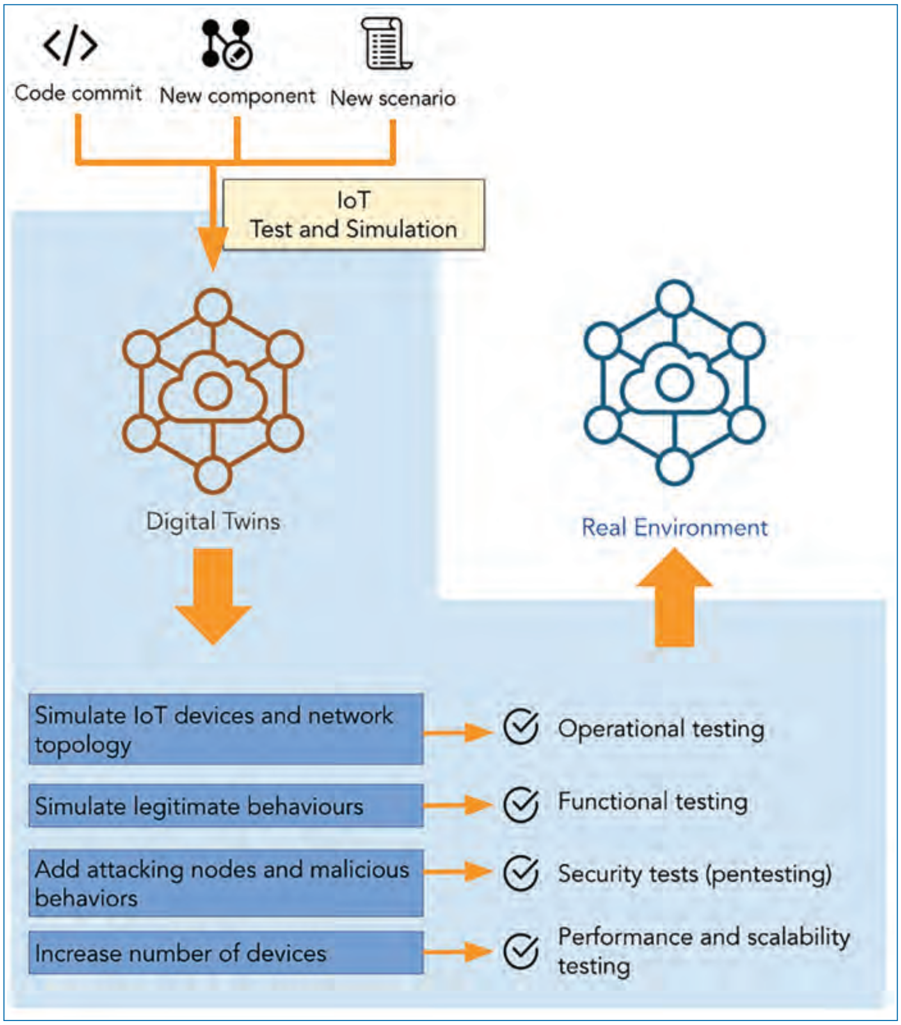

Code commit
New component
New scenario
IoT
Test and Simulation
Digital Twins
Real Environment
Simulate IoT devices and network topology
Operational testing
Simulate legitimate behaviours
Functional testing
Add attacking nodes and malicious behaviors
Security tests (pentesting)
Increase number of devices
Performance and scalability testing

Figure 8.17. Test and Simulation (TaS) Workflow.

connect devices. A test case is a collection of tests executed on one network topology. There can be many different testing types (e.g., functional, security, or scalability testing). A dataset defines a test. For test execution, the TaS enabler runs the simulation and testing using each dataset by following the test order in the test list. We can change the order of the datasets via the web interface.

Test campaign and the integration into DevOps cycle

While the test case groups the test by the defined network topologies, the test campaign contains all the test cases that should be executed for each change in the IoT system. The test campaign is the global test that covers every testing scenario and testing aspect. The test campaigns are executed automatically every time there is

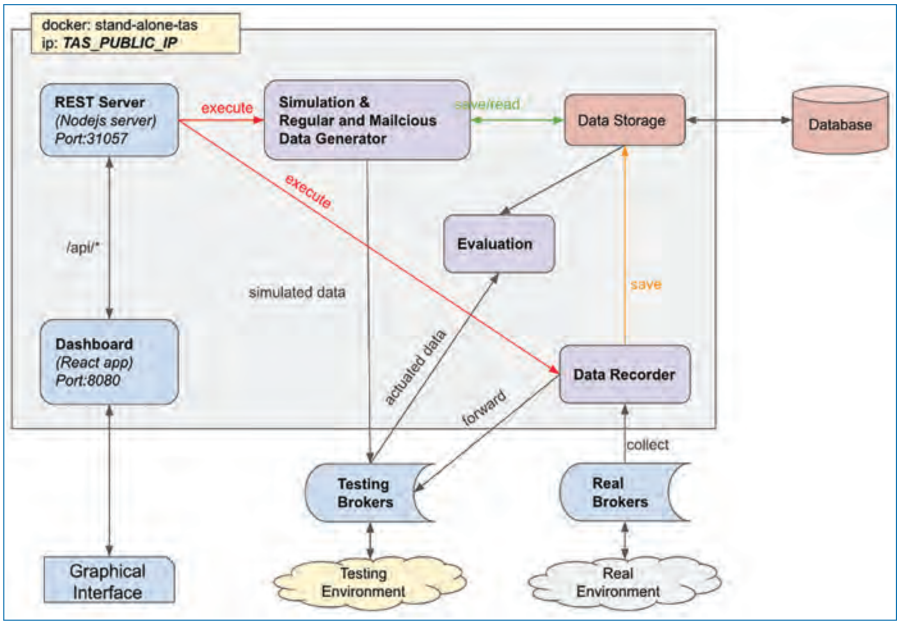


Figure 8.18. Test and Simulation (TaS) Enabler Docker image.

a change in the system. For each test campaign, the TaS enabler runs test cases according to the order in the list.

8.2.3.6 The evaluation module

The Evaluation module collects the simulated actuator data as well as the other metrics of the system. Then, it performs the evaluation based on the testing methodology (see Section 8.2.3.1 for more details).

The next section presents the implementation of the TaS enabler.

8.2.4 Implementation

8.2.4.1 The test and simulation (TaS) docker image

The TaS enabler has been designed to be portable. It can be installed as a Node.js application and packaged as a docker image. Figure 8.18 presents the communications between the modules inside a docker container and between the docker container and other modules.

The REST Server provides an API to interact with the tool. Via this API, we can execute the module Data Recorder, Simulation, and Regular and Malicious Data Generator. The Database is external to the docker container and can be connected via the Data Storage module. The dashboard is the graphical interface implemented using ReactJS [27].

Table 8.1. Basic APIs to integrate into a DevOps cycle.

Path	Method	Data	Response
/devops/	GET		Get automation testing configuration
/devops/	POST	{webhookURL, testCampaignId}	Update the automation testing configuration
/devops/start	GET		Trigger the simulation and testing process
/devops/stop	GET		Stop the simulation and testing process
/devops/status	GET		Get the status of the current execution

8.2.4.2 Basic APIs

Table 8.1 presents the list of basic APIs exposed by the tool for integration into a DevOps cycle.

8.2.5 Evaluation

The TaS enabler has been evaluated in several use cases in the ENACT project.

8.2.5.1 Itelligent train system

Figure 8.19 shows the data flow of the TaS enabler in the Intelligent Train System (ITS) use case. The Data Recorder records the WSN Coordinator data from broker-01 part of the ITS system. The recorded data is stored in the Data Storage. The Simulated Wire Sensor Network (WSN) Coordinator uses the recorded data to simulate a several WSN Coordinators for testing the scalability of the ITS system. All the simulated WSN Coordinators publish the data to broker-02 which is in the ITS system under test. By evaluating the gateway status, we can assess the scalability of the ITS system.

8.2.5.2 E-Health system

The TaS enabler is used in an e-Health use case to test if the parsing data function works correctly in the e-Health gateway. Figure 8.20 presents the data flow of the e-Health use case. The simulated sensors send some valid and invalid data messages to the internal broker, and then these messages are consumed by the CloudAgent. By mutating data messages, we can test the CloudAgent in various test scenarios, such as "invalid data format" and "invalid value".

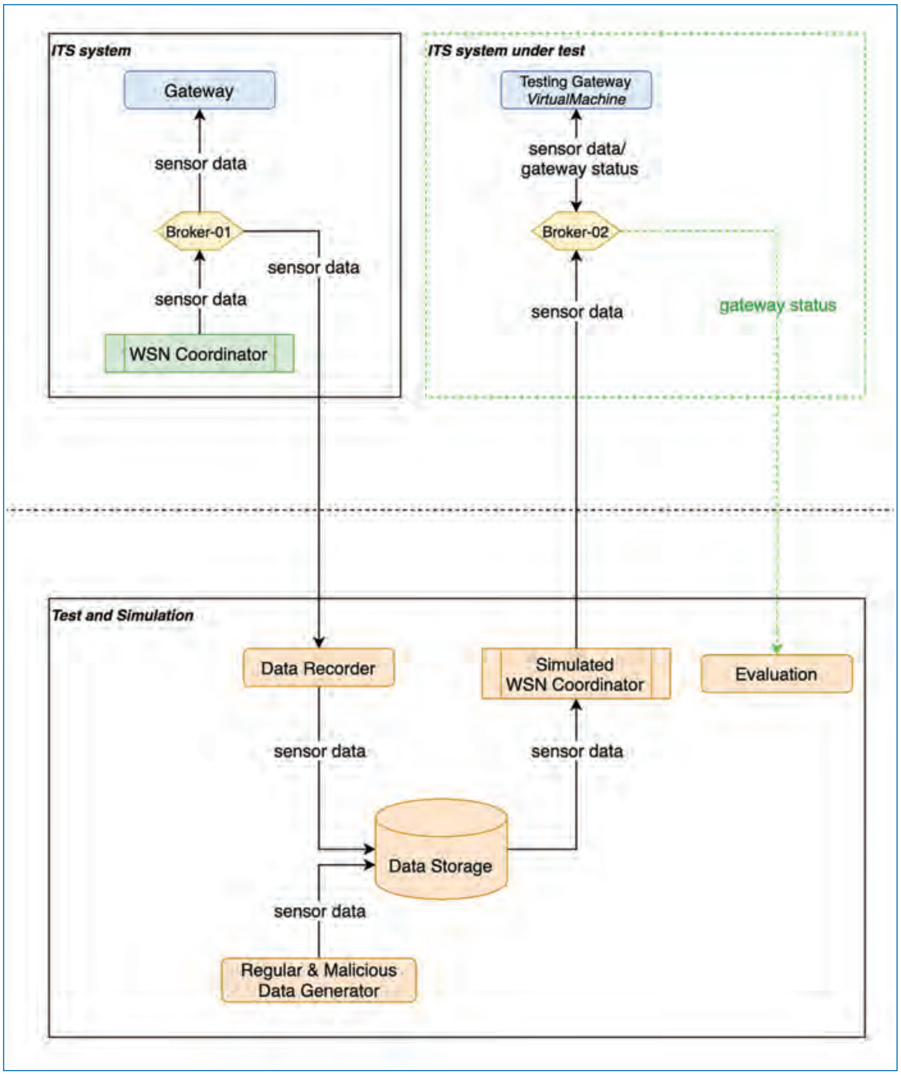


Figure 8.19. Intelligent Train System.

8.2.5.3 Smart home system

In the Smart Home use case, we used the TaS enabler as part of the DevOps cycle. Figure 8.21 shows the data flow of the TaS enabler. First, the Data Recorder records and builds the testing dataset from the real system. For each system change, such as new features, and software updates, the TaS enabler uses the recorded dataset to check the system reaction. By comparing the recorded system output with the simulated system output, we can detect miss behaviors in the updated system.

Using the TaS enabler, we can automatically test the SIS in various test scenarios. However, due to the SIS complexity, a problem may happen at any moment while

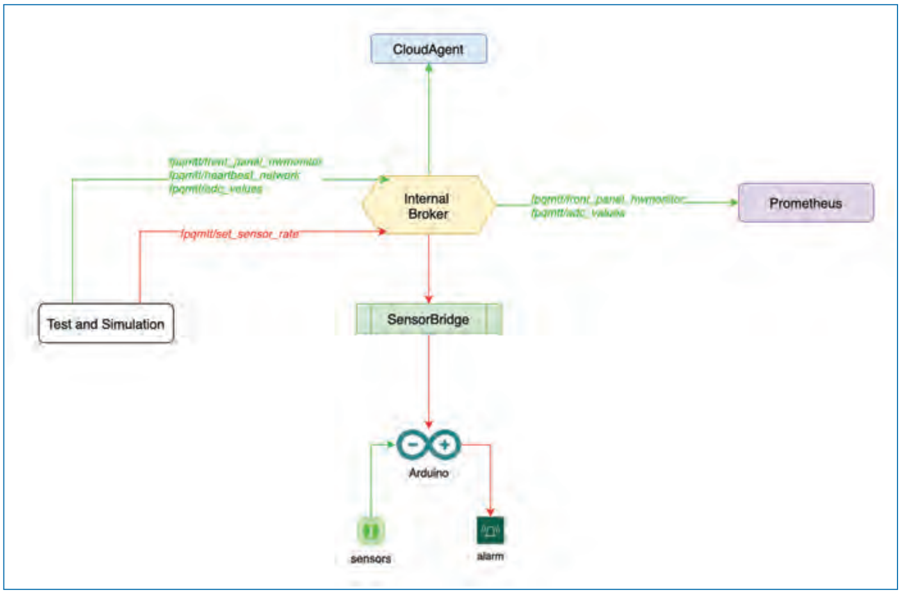


Figure 8.20. E-Health System.

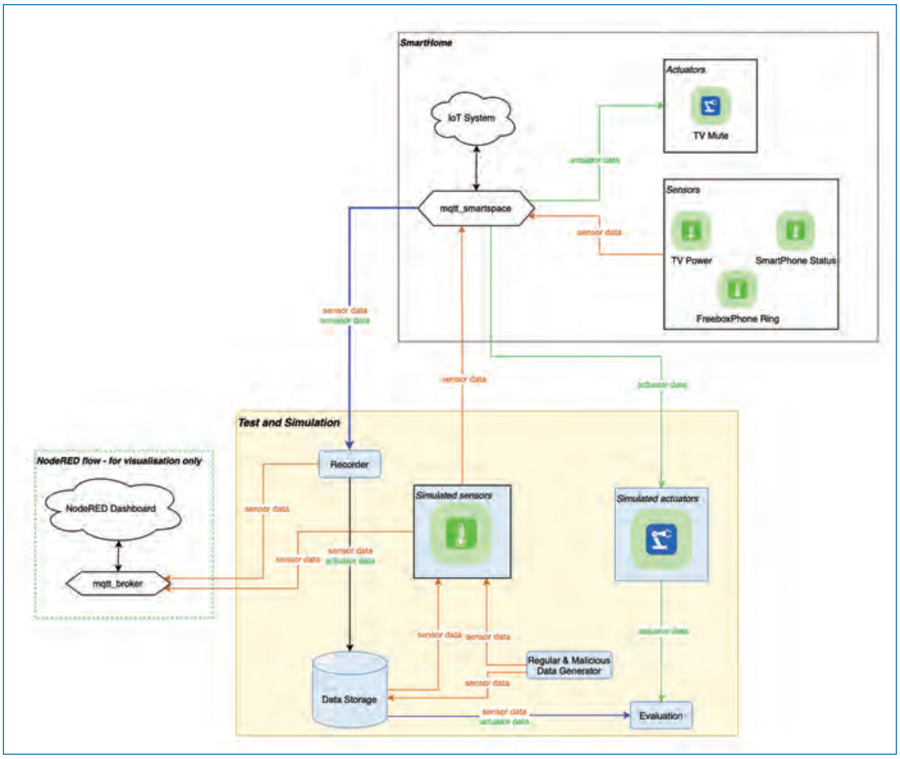


Figure 8.21. Smart Home System.

the system is running. Knowing the root cause of the problem is very important to find the solution. Therefore, we need a tool to identify the root cause of the problem while running an SIS.

8.3 Root-Cause Analysis (RCA)

Root Cause Analysis (RCA) is a systematic process for identifying "root causes" of problems or events and for responding to them. System administrators and DevOps engineers use RCA not only for detecting the problems but also for understanding their root-causes to prevent the recurrences and/or mitigate the impact. In the context of ENACT, the RCA enabler relies on Machine Learning algorithms to identify the most probable cause(s) of detected anomalies based on the knowledge of similarly observed ones. Figure 8.22 presents the high-level architecture of the implemented enabler.

The **data collector** allows gathering information from different sources (e.g., network, application, system, hardware) by relying on dedicated monitoring agents. It has a plugin architecture that enhances its extension to new data formats. Parsing such data allows extracting various attribute values relevant to the origin of any detected incident. We automatically select the most relevant attributes by using several machine learning algorithms. These attributes increase the analysis accuracy and reduce the data dimensions as well as the computation resources needed.

The **historical data** is a set of data used for learning purposes. It consists of labeled records collected over time. These records describe the original cause of several incidents (e.g., a sensor is no longer permitted to send data to the central gateway) and the relative attribute values (e.g., downstream data bit-rate measured

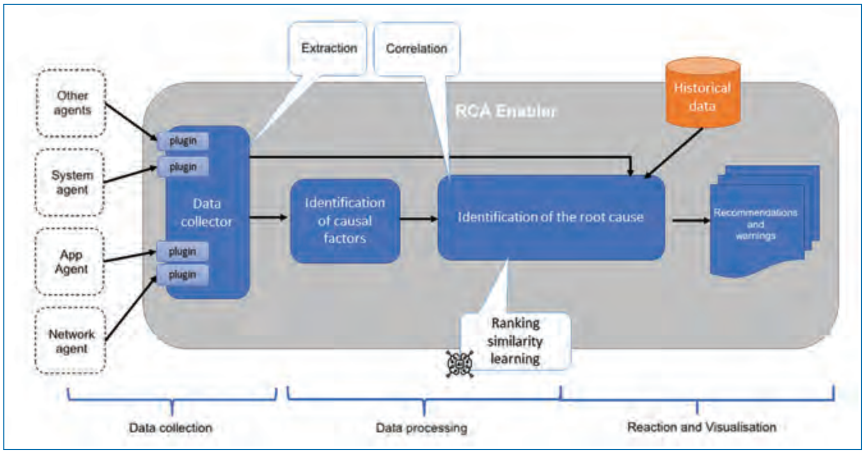


Figure 8.22. RCA Enabler high level architecture.

in the central gateway decreased). The **historical data** is constructed by two means:

- Active learning: We deal with controlled systems. Therefore, the collected data can be easily labeled by actively performing different tests injecting known failures and attacks.
- Passive learning: Once an incident is detected and alerted (e.g., by a third-party tool) without knowing its origin, thanks to the aid of the system experts, manual RCA is performed by debugging different logs and correlating various events to determine the corresponding root causes. The result of this work can be stored in the database with its relevant attribute values.

The historical data are derived from these two sources. The idea is to determine when the system reaches a known undesirable state with a known cause. It involves using the concept of Similarity Learning [8], i.e., Ranking Similarity Learning. The RCA tool calculates the similarity of the new state with the known ones. It presents the most similar states in the relative similarity order. The final goal is to recognize the incident's root origin by using historical data. In this way, the tool can recommend to the operator which countermeasures to perform based on known mitigation strategies.

The RCA Enabler works following two phases: the knowledge acquisition phase (Figure 8.23) and the monitoring phase (Figure 8.24). The former is for building a historical database of known problems and incidents. The latter consists of monitoring the system in real-time, analyzing the newly-coming incident by querying the historical data, and suggesting possible root causes. It is worth noting that passive learning in the knowledge acquisition phase can be continuously run during the monitoring phase. We describe the details of each module in the following subsections.

8.3.1 Data Collection

Analyzing an SIS requires different statistics and data, i.e., the logs, metrics, network traffic, and any data that could identify the system state. A data collector is necessary and can be provided by the system (e.g., in the ITS use case, the metrics are collected and sent to the RCA enabler via the MQTT broker), or an enabler can be deployed to collect different types of data, namely:

- Capturing network traffic: For example, the MMT-Probe [11] (TCP/IP networks) and the MMT-IoT [4] (IoT- 6LoWPAN) are able to sniff and record the network traffic.

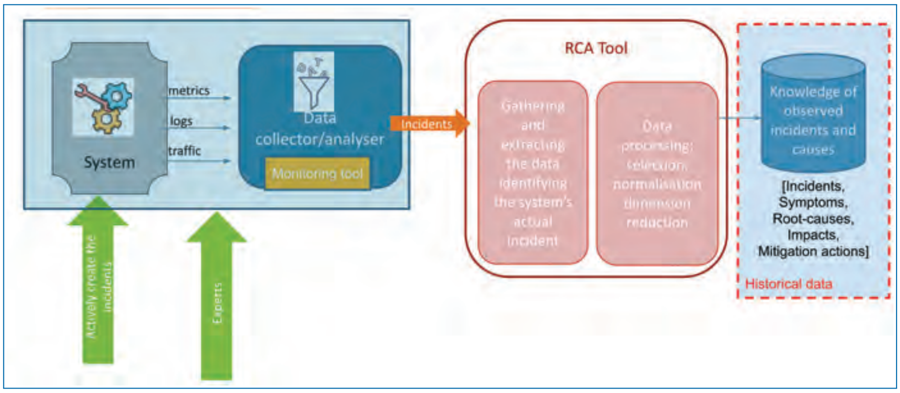

System
metrics
logs
traffic
Data collector/analyser
Monitoring tool
Incidents
RCA Tool
Gathering and extracting the data identifying the system's actual incident
Data processing: selection, normalisation, dimension reduction
Knowledge of observed incidents and causes
[Incidents, Symptoms, Root-causes, Impacts, Mitigation actions]
Historical data
Actively create the incidents
Experts

Figure 8.23. RCA-Knowledge acquisition phase.

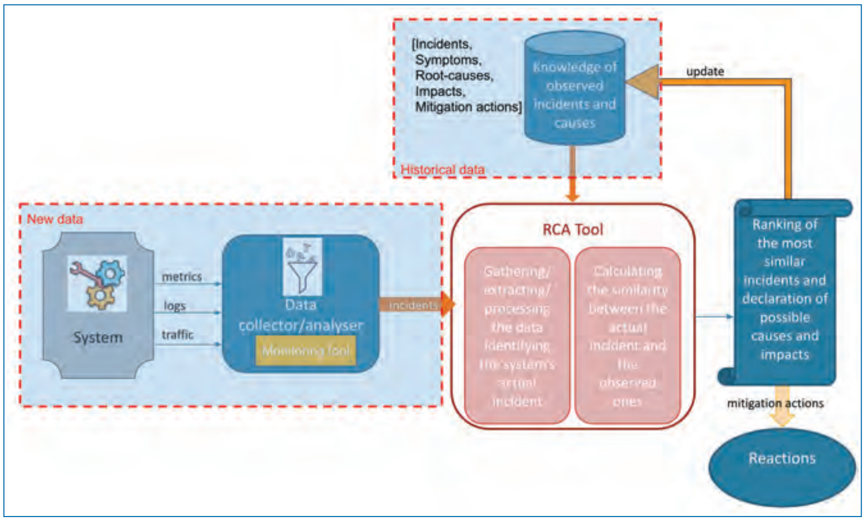

[Incidents, Symptoms, Root-causes, Impacts, Mitigation actions]
Knowledge of observed incidents and causes
Historical data
update
New data
System
metrics
logs
traffic
Data collector/analyser
Monitoring tool
incidents
RCA Tool
Gathering/ extracting/ processing the data identifying the system's actual incident
Calculating the similarity between the actual incident and the observed ones
Ranking of the most similar incidents and declaration of possible causes and impacts
mitigation actions
Reactions

Figure 8.24. RCA-Monitoring phase.

- Reading and extracting logs: The current version of the RCA enabler supports by default reading the data input in the form of JSON and CSV files. Other formats can be rapidly taken into account thanks to the extensibility of MMT-Probe (e.g., creating new plugins).

In the knowledge acquisition phase, the data can be collected in two ways:

- Actively injecting or reproducing known failures and attacks, then collecting labeled corresponding data.
- Passively monitoring the system, debugging different logs and traces, correlating various events and particularly by consulting the system experts to determine the corresponding root causes as well as the relevant data.

In the monitoring phase, the data is collected and transmitted to the RCA enabler in real-time. In theory, there is no restriction in the type of data to be gathered. On the contrary, a maximum of data for identifying the system functionalities is desirable. Even though some data could be redundant, data processing steps are performed to extract the most pertinent data.

8.3.2 Data Processing

As we mentioned in Section 8.1, there can be an enormous number of components/indicators in the analysis of an IoT system. In the following subsections, we discuss our techniques that avoid data noise, deal with heterogeneous data, and calculate the similarity between two different data sets.

8.3.2.1 Attribute selection

Attribute selection (also known as feature selection) [5] is one of the core concepts in Machine Learning that tremendously impacts the model performance. For complex systems, it is common that the data collected is too complicated or redundant. In other words, there might be some irrelevant or less important attributes (i.e., noises) contributing less to the target variable. Removing the noises helps not only to improve the accuracy but also to reduce the training time. It is the first and most essential step that should be performed automatically based on the feature selection techniques or manually by system experts.

The current version of the RCA enabler has been integrated with the following feature selection techniques:

- Univariate feature selection: The selection of the best features is based on univariate statistical tests. Each feature is compared to the target variable while the other features are temporarily ignored. The goal is to determine whether there is any statistically significant relationship between them. Each feature has its test score. The bigger the score is, the more likely the feature is important. The features with top scores should be selected. The test score is the average of the scores calculated based on the chi-square test, the f test, and the mutual information classification test [5].
- Recursive feature elimination (RFE): It is about selecting features by recursively considering smaller and smaller sets of features. The idea is to use an external estimator (logistic regression model and random forest model [7]) that assigns weights to features (e.g., linear model coefficients). The least important features are pruned step-by-step from the current set of features. This procedure is recursively repeated on the pruned set until the desired number of features left is eventually reached. Compared to univariate feature selection, RFE considers all features at once, thus can capture interactions.

8.3.2.2 Data normalisation

Normalization is a concept informally used in statistics, and the term "normalized data" may have different meanings. In principle, data normalization means eliminating heterogeneous data measurement units and making the attributes comparable despite different value ranges.

In our perspectives, data normalization consists of two steps:

- Standardizing data to have a **mean** of zero and a **standard deviation**(s) of 1 (Equation (8.1) and Figure 8.25):

$$x_{standardized} = \frac{x - mean(x)}{s} \tag{8.1}$$

- Re-scaling the data to have values between 0 and 1 (Equation (8.2)):

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{8.2}$$

8.3.2.3 Similarity calculation

Suppose that the system's temporal state can be reflected by n metrics (i.e., n attributes). This set of n attributes can be represented by a vector in a multi-dimensional space of n dimensions. Calculating the similarity and dissimilarity of two states becomes the problem of measuring the distance of orientation (the angle) and magnitude (the length) of their two representing vectors. Figure 8.26 depicts an example in a 3-dimensional space.

The current version of the RCA enabler has been integrated with the following similarity and distance measures:

- Cosine similarity [2]
- Adjusted cosine similarity [2]
- Jaccard similarity [2]
- Euclidean distance [2]
- Manhattan distance [2]
- Minkowski distance [2]

These measures are used to calculate the similarity score whose value is between 0 and 1. The bigger the similarity score is, the more similar the two compared states are (e.g., if the similarity score is equal to 0.95, there is a 95% probability that two compared states are considered the equivalent). The similarity score can be computed based on one or multiple similarity and distance measures. In the training phase, we determine the measures. Therefore, when we compare a known

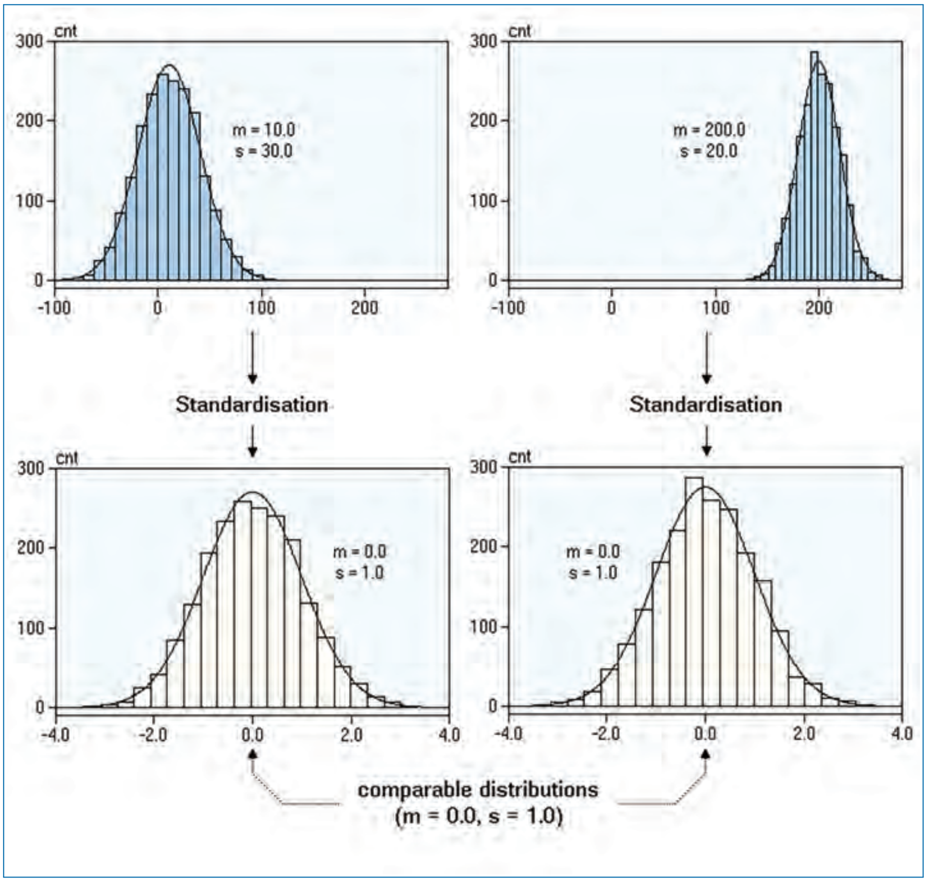


Figure 8.25. Data standardization.

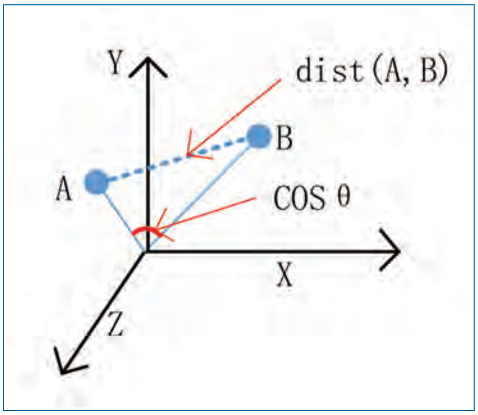
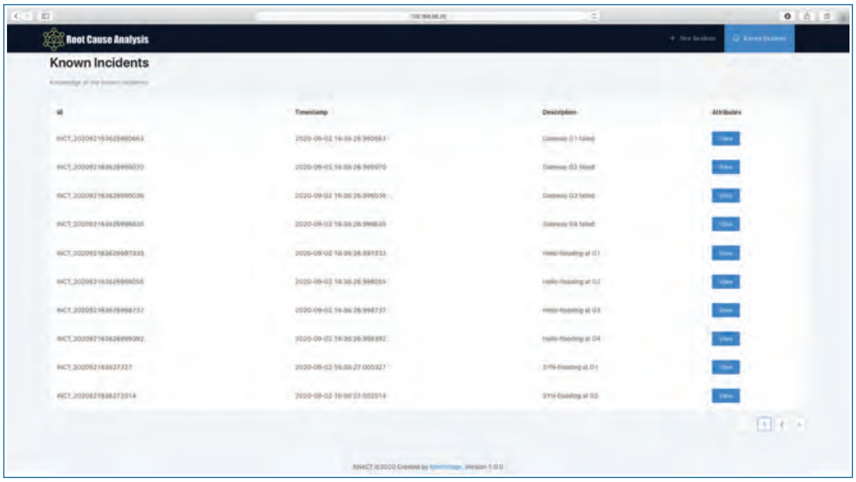


Figure 8.26. Similarity calculation in 3-dimensional space.



Known Incidents

Knowledge of the known incidents

Id	Timestamp	Description	Attributes
INCT_202092163626980663	2020-09-02 16:36:26.980663	Gateway 01 failed	View
INCT_202092163626995070	2020-09-02 16:36:26.995070	Gateway 02 failed	View
INCT_202092163626996036	2020-09-02 16:36:26.996036	Gateway 03 failed	View
INCT_202092163626996835	2020-09-02 16:36:26.996835	Gateway 04 failed	View
INCT_202092163626997333	2020-09-02 16:36:26.997333	Hello-flooding at 01	View
INCT_202092163626998055	2020-09-02 16:36:26.998055	Hello-flooding at 02	View
INCT_202092163626998737	2020-09-02 16:36:26.998737	Hello-flooding at 03	View
INCT_202092163626999392	2020-09-02 16:36:26.999392	Hello-flooding at 04	View
INCT_202092163627327	2020-09-02 16:36:27.000327	SYN-flooding at 01	View
INCT_2020921636272014	2020-09-02 16:36:27.002014	SYN-flooding at 02	View

1 2 >

EMACT ©2020 Created by Montimage. Version 1.0.0

Figure 8.27. Historical database of known incidents.

state and its repetition, we have a similarity score as big as possible. Besides, to avoid false positives, the similarity between a common proper state and each known malicious state should be as low as possible.

8.3.3 Reaction and Visualization

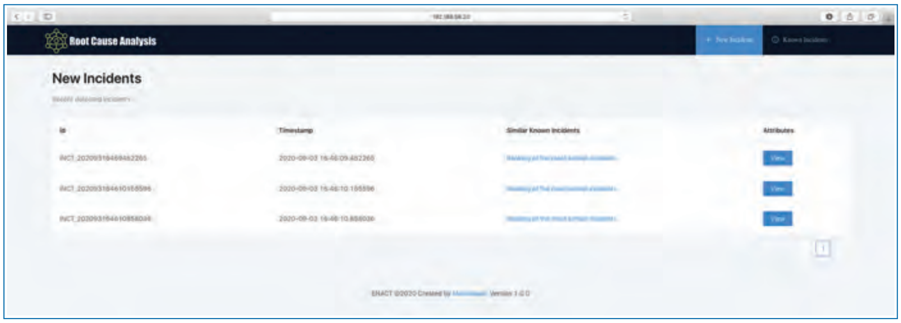
First, the RCA enabler analyses live the data originating from the system under monitoring. It reports back the similarity score of the current state, its most similar known incident, and the corresponding root causes. If the similarity score is higher than a given threshold, an alert is generated. The alert helps the system administrators foresee how the system evolves from a normal functioning state to a known fault or failure and determine the root causes to perform the most appropriate mitigation actions. For example, in the ITS use case, the RCA enabler communicates with the ITS through an MQTT connection. When the RCA enabler receives a message with all the relevant attributes, it identifies the level at which the system state and a known incident are similar. It publishes the result to the MQTT exchange.

Regarding the visualization, the results of the RCA enabler can be viewed intuitively on a GUI. Figure 8.27, Figure 8.28, Figure 8.29 displays some screenshots of RCA's GUI. More results are presented in the following sections.

8.3.4 Evaluation

8.3.4.1 Performance evaluation with generated testing data

To evaluate the RCA enabler, we first generated a learning data set in CSV format with several known records. Each record describes an "incident" with different


Root Cause Analysis

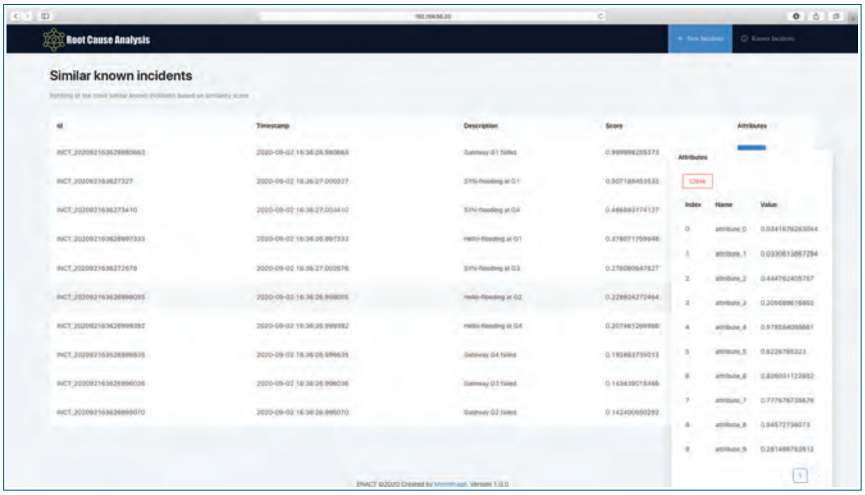
New Incidents

[illegible]

Id	Timestamp	Similar Known Incidents	Attributes
INCT_20200316460948226(5)	2020-09-03 16:46:09.482265	[illegible]	View
INCT_20200316461015559(6)	2020-09-03 16:46:10.155596	[illegible]	View
INCT_20200316461085803(6)	2020-09-03 16:46:10.858036	[illegible]	View

ENACT ©2020 Created by [illegible] Version 1.0.0

Figure 8.28. Newly detected incidents.


Root Cause Analysis

Similar known incidents

Ranking of the most similar known incidents based on similarity score

Id	Timestamp	Description	Score
INCT_20200921636269806 63	2020-09-02 16:36:26.980663	Gateway G1 failed	0.999998285373
INCT_2020092163627327	2020-09-02 16:36:27.000327	SYN-flooding at G1	0.507188453532
INCT_2020092163627341 0	2020-09-02 16:36:27.003410	SYN-flooding at G4	0.466893174127
INCT_20200921636269973 33	2020-09-02 16:36:26.997333	Hello-flooding at G1	0.378071759948
INCT_2020092163627267 6	2020-09-02 16:36:27.002676	SYN-flooding at G3	0.276080647827
INCT_20200921636269980 55	2020-09-02 16:36:26.998055	Hello-flooding at G2	0.228924272464
INCT_20200921636269993 92	2020-09-02 16:36:26.999392	Hello-flooding at G4	0.207461266986
INCT_20200921636269966 35	2020-09-02 16:36:26.996635	Gateway G4 failed	0.192863735013
INCT_20200921636269960 36	2020-09-02 16:36:26.996036	Gateway G3 failed	0.143439015466
INCT_20200921636269950 70	2020-09-02 16:36:26.995070	Gateway G2 failed	0.142400650292

Attributes

Close

Index	Name	Value
0	attribute_0	0.0341676263044
1	attribute_1	0.0330613867294
2	attribute_2	0.444762405787
3	attribute_3	0.205688616803
4	attribute_4	0.578584098861
5	attribute_5	0.8226785323
6	attribute_6	0.826031122882
7	attribute_7	0.777676735676
8	attribute_8	0.84572736073
9	attribute_9	0.281498783812

ENACT ©2020 Created by [illegible] Version 1.0.0

Figure 8.29. A newly detected incident and its similarity scores in comparison with known ones.

"attribute" values. These learned "incidents" are stored together with the potential origin causes. The "attributes" refer to the metric values that can be gathered. Afterward, we generate new attributes and check whether the system recognizes them as a known incident among the ones that are already in the historical data. The RCA enabler measures the similarity of each new record and each learned incident. It ranks them by determining the most likely similar ones. To assess the performance, we calculate the enabler's response time with the function taking as input the number of known states and the attributes identifying a state. In this evaluation, the similarity score is the average of all similarity measures mentioned in Section 8.3.2.3.

Figure 8.30 and Figure 8.31 show the results of our experiments. As expected, more time is needed when the data volume handled increases. However, we observe that the number of known states has less impact on the response time than the

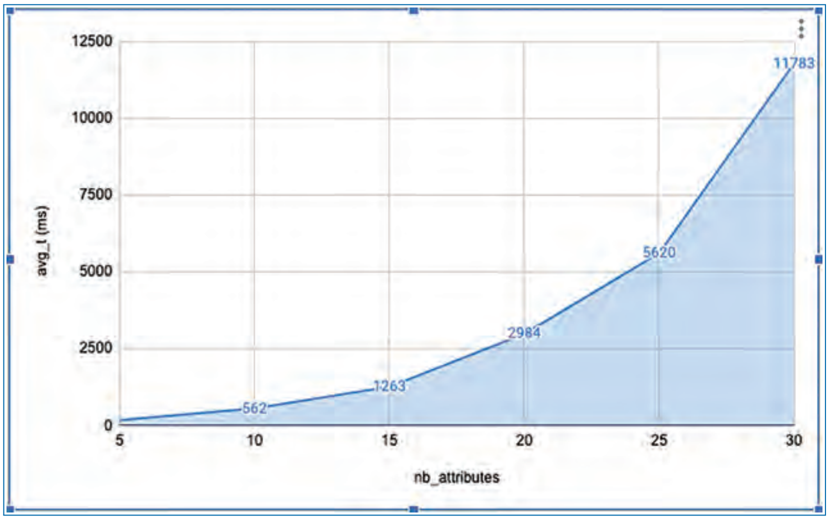


Figure 8.30. RCA Enabler's response time towards the number of attributes.

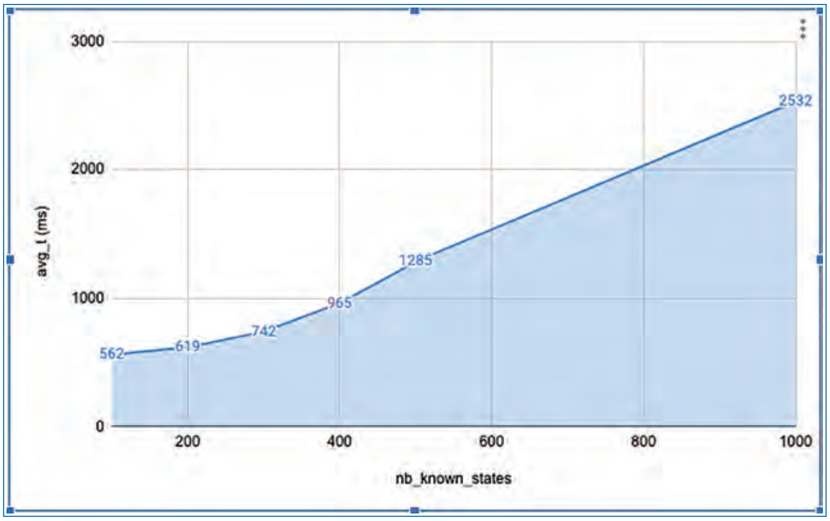


Figure 8.31. RCA Enabler's response time towards the number of known incidents.

number of the attributes has. The processing time needed increases more drastically when more attributes are under consideration than when there are more known states. This increase reaffirms the need for integrating "attribute selection" as aforementioned in Section 8.3.2.1. In our evaluation, we did not apply any selection technique because the data was generated randomly. The attributes are, thus, seemingly equal and make the selection not useful. However, there will probably be a different story when real systems are involved (further discussed in Section 8.3.4.2).

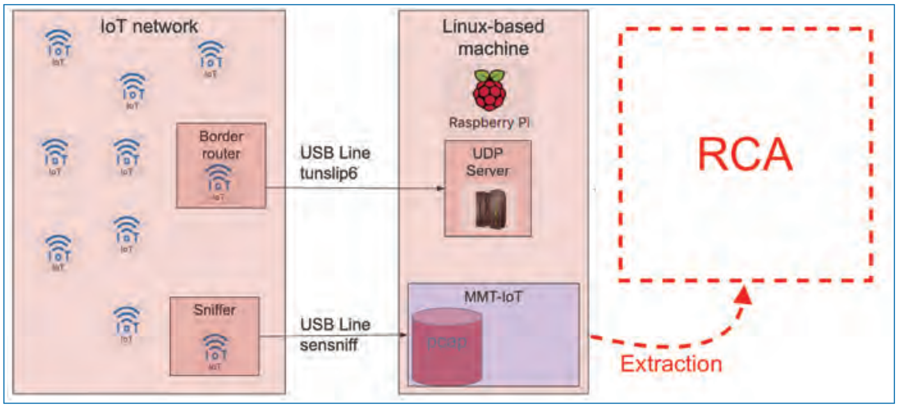


Figure 8.32. General architecture for the evaluation.

8.3.4.2 Evaluation on a real IoT Testbed

Set-up of the experiment

To evaluate the RCA enabler, we performed several experiments on a real IoT testbed called w-iLab.t[1] provided by Imec.[2] Figure 8.32 presents the general architecture of the experiments. Several IoT devices (Zolertia Re-Mote[3]) formed an IoT network where the clients reported sensed data periodically to the border router before these reports were forwarded via a USB line to a server installed in a more powerful Linux-based machine. There was an IoT device to perform sniffing tasks: capturing network traffic and piping via the USB line to the Linux-based machine where MMT-IoT was deployed to analyze the traffic and extract the metrics for the RCA enabler. Besides, the IoT network consisted of normal clients reporting sensed data every 10 seconds, and one (or several) attacker(s) behaved interchangeably in three modes:

- Normal mode reporting data every 10 seconds.
- DoS (Denial of Service) attack mode reporting data 100 times faster (10 messages/s) and with incorrect Frame Check Sequence (FCS[4]).
- Dead mode not reporting data at all (node failure).

1. https://www.fed4fire.eu/testbeds/w-ilab-t/
2. https://www.imec-int.com/
3. https://zolertia.io/zolertia-platforms/
4. FCS: The FCS field contains a number calculated by the source node based on the data in the frame. This number is added to the end of a frame sent. When the destination node receives the frame, the FCS number is recalculated and compared with the number sent in the frame. If they are different, the frame is considered malformed (intentionally or not) or modified between the source node and the destination node.

Table 8.2. Attributes extracted and sent to RCA.

Ref.	Attribute	Description
(1)	Network throughput (bps, pps)	The whole network traffic throughput, computed in bits per second (bps) and packets per second (pps)
(2)	Throughput at devices (bps, pps)	Throughput estimated at each device.
(3)	Traffic transmitted on links (bytes, packets)	Traffic volume transmitted on each link during a parameterized period (e.g., 10 seconds)
(4)	Number of routing-related packets (packets)	Number of routing-related packets sent and received by each device during a parameterized period (e.g., 10 seconds)
(5)	Transmission delay (ms)	The duration in millisecond since the packet is created by a device (timestamp packaged in the sensed data) until it is captured by the sniffer (captured packet's timestamp)
(6)	CPU usage (%)	CPU usage at each device, packaged in the sensed data.
(7)	Memory usage (%)	Memory usage at each device, packaged in the sensed data.
(8)	Battery level (%)	Level of battery left at each device, packaged in the sensed data.
(9)	Power consumption (W)	Power consumption (DC) at each device in Watts, packaged in the sensed data.
(10)	Average packet size (bytes)	Average size of packets transmitted during a parameterized period (e.g., 10 seconds)
(11)	Probe ID	An integer number representing the ID of the MMT-Probe analysing the traffic and performing the extraction.
(12)	Protocol ID	An integer number representing the protocol ID

Table 8.2 summarizes the attributes extracted by MMT-IoT and transferred to RCA for further analysis.

Attribute selection and data normalisation

As the first step, the RCA enabler selects the significant attributes among the 12 listed in Table 10. The selection is done by applying the techniques aforementioned in Section 8.3.2.1.

An incorrect FCS can signify a malformed packet (e.g., due to a misconfiguration or an error in the implementation), a jamming attack (i.e., the attacker abuses the network by generating frames that should be ignored), or a message manipulation attack (i.e., the attacker intercepts and modifies a frame's content).

Table 8.3. Feature Selection results.

	Univariate feature selection			Recursive feature elimination	
Ref.	Chi-square test	f-test	Mutual information classification test	Logistic regression model	Random forest model
(1)	true	true	true	1	true
(2)	true	true	true	2	true
(3)	true	true	true	2	true
(4)	true	false	true	6	false
(5)	true	true	true	3	true
(6)	true	true	true	2	true
(7)	true	true	true	2	true
(8)	false	false	false	9	false
(9)	false	true	true	6	false
(10)	false	false	false	8	false
(11)	false	false	false	10	false
(12)	false	false	false	10	false

Table 8.3 summarizes the results when different Feature Selection models are used. There are six attributes, namely (1–3), (5–7), which are considered significant according to all the models. Four attributes (8), (10), (11), and (12) are concluded to be not relevant and can be left out. The attributes (4) and (9) are recommended by some models and not by others. We performed the analysis in the following subsection with these two attributes and the other six attributes recommended by all the models.

Similarity calculation and analysis

Firstly, regarding the DoS attack, one can see clearly in the statistics displayed by MMT-IoT that:

- The traffic volume increased significantly during the attack period (Figure 8.33).
- The attacker was evidently the most active device (Figure 8.34) and one end of the most active link (Figure 8.35).

From the RCA point of view, all other selected attributes were more or less affected by the DoS attack. The attack pattern was learned, and when repeated,

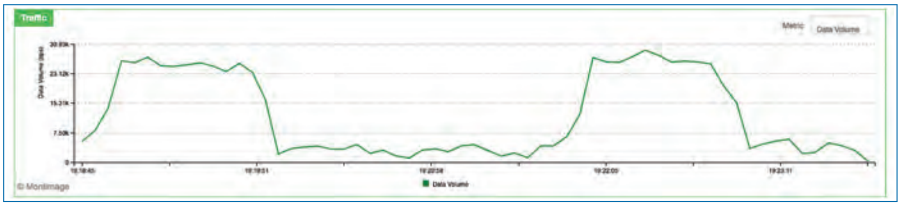
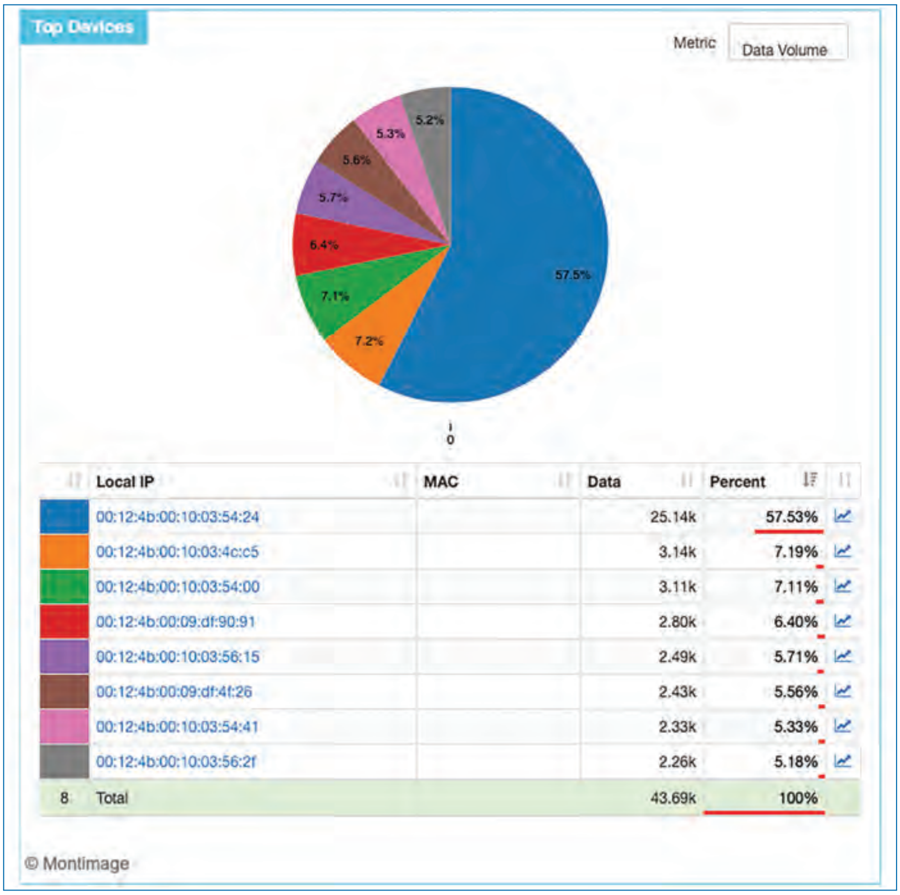


Figure 8.33. Traffic throughput increased remarkably when the attack took place.

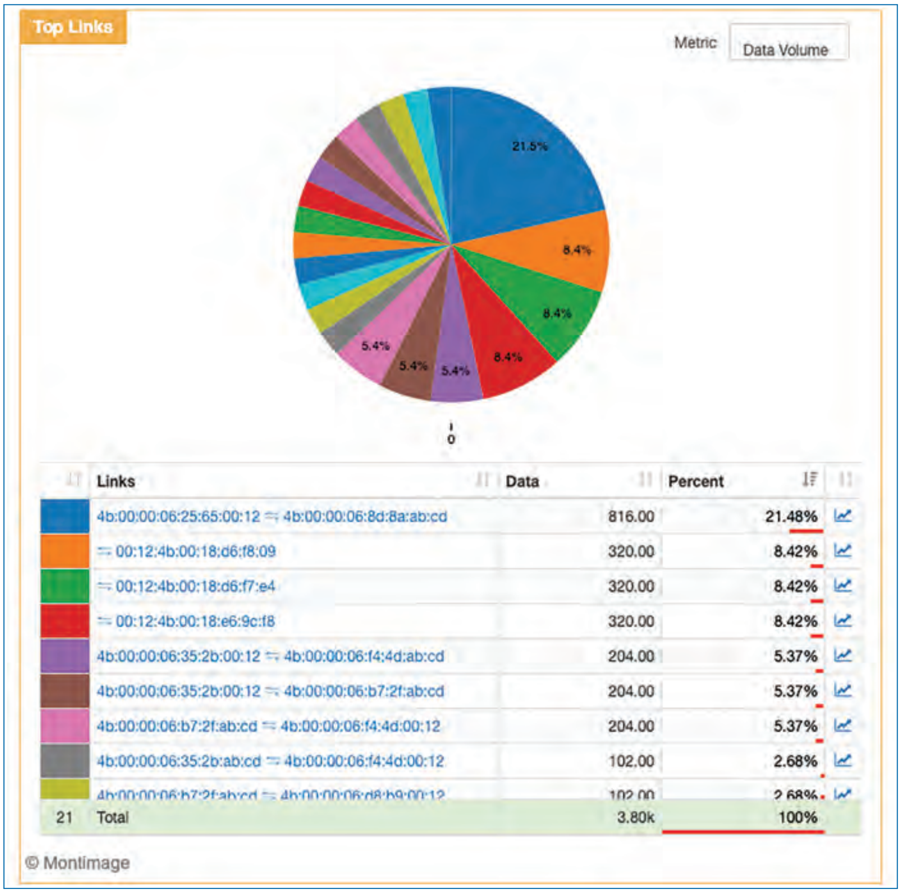


	Local IP	MAC	Data	Percent
🟦	00:12:4b:00:10:03:54:24		25.14k	57.53%
🟧	00:12:4b:00:10:03:4c:c5		3.14k	7.19%
🟩	00:12:4b:00:10:03:54:00		3.11k	7.11%
🟥	00:12:4b:00:09:df:90:91		2.80k	6.40%
🟪	00:12:4b:00:10:03:56:15		2.49k	5.71%
🟫	00:12:4b:00:09:df:4f:26		2.43k	5.56%
▨	00:12:4b:00:10:03:54:41		2.33k	5.33%
⬜	00:12:4b:00:10:03:56:2f		2.26k	5.18%
8	Total		43.69k	100%

Figure 8.34. The attacker was the most active device.

the similarity score observed by the RCA was always no less than 0.92 (i.e., 92% similar). It is worth noting that, in this evaluation, we computed the similarity score based on the "adjusted cosine similarity".

In node failure (dead device), all the attributes related to the dead device were affected. The RCA enabler reported a similarity score between 0.84 and 0.87 when the failure repeated.



	Links	Data	Percent	
🟦	4b:00:00:06:25:65:00:12 ⇆ 4b:00:00:06:8d:8a:ab:cd	816.00	21.48%	📈
🟧	⇆ 00:12:4b:00:18:d6:f8:09	320.00	8.42%	📈
🟩	⇆ 00:12:4b:00:18:d6:f7:e4	320.00	8.42%	📈
🟥	⇆ 00:12:4b:00:18:e6:9c:f8	320.00	8.42%	📈
🟪	4b:00:00:06:35:2b:00:12 ⇆ 4b:00:00:06:f4:4d:ab:cd	204.00	5.37%	📈
🟫	4b:00:00:06:35:2b:00:12 ⇆ 4b:00:00:06:b7:2f:ab:cd	204.00	5.37%	📈
■	4b:00:00:06:b7:2f:ab:cd ⇆ 4b:00:00:06:f4:4d:00:12	204.00	5.37%	📈
■	4b:00:00:06:35:2b:ab:cd ⇆ 4b:00:00:06:f4:4d:00:12	102.00	2.68%	📈
■	4b:00:00:06:b7:2f:ab:cd ⇆ 4b:00:00:06:d8:b9:00:12	102.00	2.68%	📈
21	Total	3.80k	100%	

Figure 8.35. The attacker belonged to the most active link.

Lastly, when the border router was affected by the jamming attack of incorrect FCS values, its CPU usage jumped virtually. When this behavior happened again, the RCA enabler determined that it was up to 94% similar to the learned incident's observed symptoms. However, even when all the network devices worked normally, the RCA enabler identified that there was up to 78% similarity between this incident and the event "Possible jamming attack with incorrect FCS values".

8.4 Conclusion

In conclusion, This chapter presents two tools, i.e., the TaS and RCA enablers, that enable the validation and verification of IoT systems. The TaS enabler, based on the idea of "Digital Twins", is a Software-as-a-Service solution that provides

(i) a flexible simulation of sensor networks, (ii) a powerful data generator with real-time data recording, and (iii) support for Continuous Integration and Continuous Development. It helps IoT application developers save time and money on setting up the testing environment and thus supports faster application delivery. The RCA enabler systematizes the knowledge about the potential incidents that may occur in the system. It prevents the incidents or to quickly and intelligently react against their recurrences.

In practice, we can apply the TaS and RCA enablers to various systems other than IoT systems in the context of ENACT. In general, they can work on any system in which the data about the system's functioning state can be collected. For the RCA enabler, it would be beneficial if the owner or administrator has already acquired a certain level of understanding about the system to facilitate the training phase and the database creation for known incidents and root causes. Otherwise, we can perform penetration tests to discover potential vulnerabilities so that the attacks and failures can be injected and learned.

Moreover, both tools have been developed to be generic enough so that adaptations can be easily made to make them applicable to various types of systems (e.g., industrial SCADA systems, 5G mobile networks). We plan to adapt and use these two tools in several other collaborative projects in different contexts. We hope they will play a crucial role in the Montimage ecosystem and be commercialized within the MMT Monitoring solution.[5]

References

[1] C. Adjih *et al.* "FIT IoT-LAB: A large scale open experimental IoT testbed". In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. 2015, pp. 459–464. DOI: 10.1109/WF-IoT.2015.7389098.

[2] Marc Sebban Aurélien Bellet, Amaury Habrard. *Metric Learning, Similarity-Based Pattern Analysis and Recognition*. Morgan and Claypool, 2015. ISBN: 1939-4616.

[3] A. Fuller *et al.* "Digital Twin: Enabling Technologies, Challenges and Open Research". In: *IEEE Access* 8 (2020), pp. 108952–108971. DOI: 10.1109/ACCESS.2020.2998358.

[4] Vinh Hoa La, Raul Fuentes, and Ana R. Cavalli. "A Novel Monitoring Solution for 6LoWPAN-based Wireless Sensor Networks". In: *Proceedings of 22nd Asia-Pacific Conference on Communications (APCC 2016)*. 2016.

5. https://montimage.com/products/MMT_DPI.html

[5] Richard Lowry. *Concepts and Applications of Inferential Statistics*. Vassar College, 2008.

[6] Sara N. Matheu-García *et al.* "Risk-based automated assessment and testing for the cybersecurity certification and labelling of IoT devices". In: *Computer Standards & Interfaces* 62 (2019), pp. 64–83. ISSN: 0920-5489. DOI: https://doi.org/10.1016/j.csi.2018.08.003. URL: https://www.sciencedirect.com/science/article/pii/S0920548918301375.

[7] Kjell Johnson Max Kuhn. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Taylor & Francis, 2019.

[8] Marcello Pelillo. *Similarity-Based Pattern Analysis and Recognition*. Springer, 2013. ISBN: 978-1-4471-5628-4.

[9] Luis Sanchez *et al.* "SmartSantander: IoT experimentation over a smart city testbed". In: *Computer Networks* 61 (2014). Special issue on Future Internet Testbeds Part I, pp. 217–238. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.bjp.2013.12.020.

[10] M. Tappler, B. K. Aichernig, and R. Bloem. "Model-Based Testing IoT Communication via Active Automata Learning". In: *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. 2017, pp. 276–287. DOI: 10.1109/ICST.2017.32.

[11] B. Wehbi, E. Montes de Oca, and M. Bourdelles. "Events-Based Security Monitoring Using MMT Tool". In: *IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST), 2012*. Apr. 2012, pp. 860–863. DOI: 10.1109/ICST.2012.188.

DOI: 10.1561/9781680838251.ch9

Chapter 9

SIS-based eHealth Application: The Tellu Use Case

By Arnor Solberg, Oscar Zanutto and Franck Fleurey

9.1 From Chronic to Pro-active Care

To date, an average of 80% of public health care resources in Europe are spent to respond to chronic diseases that are exacerbated in the last three years of people's lives, against a low investment of resources in the field of prevention, which can be pursued by changing lifestyles. Public health expenditure is among the largest and fastest growing spending items for governments. In 2015, public expenditure on health was 7.8% of GDP in the EU as a whole, with more than 70% of expenditure funded by the public sector in two thirds of Member States (EC 2017). In 2013, premature deaths due to major NCDs (cardiovascular diseases, cancers, respiratory diseases and diabetes) cost EU economies 0.8% of GDP (OECD/ EC 2016), with further losses incurred due to the lower productivity and employments rates of people living with chronic health problems. Due to population aging, chronic diseases and the diffusion of new diagnostic and therapeutic technologies, the share of GDP spending on health is projected to increase in the coming years (EC 2015, OECD/ EC 2016). In most high and middle-income countries, non-communicable diseases are responsible for the biggest share of such healthcare costs (EC 2014). Furthermore the ongoing pandemic situation has boosted the demand of online telecare, eHealth solutions. General practitioners and nurses have improved the remote physical parameters gathering, almost regarding oxygen saturation and body temperature. Due to these kind of increasing use of technology, care provider organizations has improved and scaled up their organizational models also in terms of digital employees' digital skills and care workflows.

This panorama indicates that institutions and care providers adopt a pro-active care approach aimed at significantly influencing individual, collective and organisational behaviour so that people can consciously plan lifestyles in which the preventive role played by the behavioural determinants of longevity is valued: nutrition, physical activity, cognitive stimulation and sociality.

This last aspect, in particular, has a decisive role as a health protection factor. It has been shown that perceived loneliness has an impact in terms of mortality comparable to smoking fifteen cigarettes a day. Furthermore, it is frequently associated with anxiety, depression and reduced movement, which can lead to hypertension and metabolic disorders with chronic degenerative effects. In this sense, living, lifestyle and technological support are integrating into a unicum that characterises the ecosystem in which the health design of the future is embedded. Within this new perspective, the process of longevity requires social and health services to overcome the dichotomous logic of intervention structured on antitheses such as health vs. disease, autonomy vs. dependence, placing rather their offer within a continuum that contemplates paths of reversibility, compensation, homeostasis, and new dynamic adaptations to the needs of the subject.

It is therefore necessary to imagine a model of person-services relationship based on the concept of co-production of health. In such a framework, the intervention of technology is inserted in support of care in a co-decided way with the person: at one extreme, ICT assumes a role of support to the fitness and well-being (i.e. prevention) of autonomous and still healthy citizens, to reach at the opposite extreme the apex of the technological complexity connected to an increase in the intensity of health care, passing through moments in which it becomes possible to set up an "intermediate" action of technological support, for example in the management of chronicity at home and in the transitions between the services used by the subject. eHealth solutions have increased their presence, and their perceived usefulness, following the development of the Covid pandemic19. In the global context, and in the European context in particular, there has been a proliferation of state-sponsored applications that provide information on the disease, ensure contact tracing, and create an informed dialogue with one's doctor and the Covid19 emergency management team in one's territory. The applications have also made it possible to remove much of the bureaucracy involved in the relationship between citizens and the health system, since many of the activities relating to the booking of diagnostic examinations, their payment and their reporting have moved online. Lastly, one of the most interesting aspects of the increase in the use of wearable technologies for measuring certain health parameters is the growing possibility of acquiring data capable of feeding machine learning and data processing systems capable of activating artificial intelligence systems capable of making diagnostic forecasts and providing useful information for personalising care in increasing numbers.

At the centre of this paradigm is the decision-making process involving experts and the person: it is based on the personalisation and timing of the use of technologies, and on the personalised design of the integrated care process structured in collaboration with the family (where present) and the social and health services, public and private, which may be activated.

9.2 e-Health and m-Health for the Digital Evolution of Services

The European Commission defines eHealth and Digital health and care as: "Digital health and care refers to tools and services that use information and communication technologies (ICT) to improve prevention, diagnosis, treatment, monitoring and management of health and lifestyle. Digital health and care has the potential to innovate and improve access to care, the quality of care and to increase the overall efficiency of the health sector".

In the field of care for the elderly, tele-assistance is one of the answers that best translates the above statement into concrete terms. It must be understood as the person's ability to communicate remotely with home care providers and their social surroundings through the use of devices such as tablets and smartphones. These devices, equipped with integrated adapted video communication applications, are often placed in dialogue with wearable devices capable of automatically acquiring information about certain significant critical parameters, such as blood pressure and blood sugar in the case of fragile people suffering from chronic diseases such as hypertension and diabetes. They are therefore able to signal the exceeding of individual critical thresholds, activating the subject and starting a pre-set alarm chain.

In this sense, in 2017 the European Commission launched the initiative "Blueprint strategy for a digital transformation of health and care in an ageing society" proposing a structured path that links four distinct but fundamental worlds in advancing care innovation alongside technology: universities, companies, public authorities and citizens. The objective pursued is to transform social challenges into opportunities for economic growth associated with an increase in citizens' wellbeing. This initiative foresees a "multiplier effect" to boost the digital transformation of the entire health care secotr. For companies, research organisations and care providers operating in the social and health sectors, the indications contained in the

Figure 9.1. Evidence of the positive results produced by the implementation of this approach, defined as "quadruple helix", is the success of numerous experiences mapped by the study of European excellence in the sites selected by the European Innovation Partnership for Active and Healthy Ageing.

"Blueprint strategy" represent a fundamental reference point for structuring digital innovation paths in care. This document directs corporate efforts towards the adoption of a perspective in which people and their needs are placed at the centre, aiming at their empowerment to achieve independent living in their own context. The elements connected to the participatory co-design of technological solutions, as well as the creation of sustainable business models capable of making care systems more efficient, represent the drivers to be followed for the digital transformation of services supporting frail persons.

A virtuous example is represented by the recent HoCare2.0 Project, done in the Interreg Central Europe Programme (https://www.interreg-central.eu/Content.Node/HoCare2.0.html) that is going to codesign and provide customer-centered home care by co-creation with citizens. The Project foresee the creation and the devices adaptation to the user needs in combination with the SMEs knowledge to come up with technological solutions that could be relevant and usable in the daily life.

Another experience is The "Electronic Health Care Record and Integrated Information Systems" that has been implemented by the Valencian Health Agency to improve the integration and interoperability of systems and guarantee their sustainability, with greater efficiency and quality of service and according to a citizen-centred approach.

Below are some examples of the impact of the programme:

- In the context of the Integrated Home Care Programme, approximately 7,000 patients were treated with an overall satisfaction index of 92.7%; 154

were saved for each stay in hospital, which was 30% less than the Spanish national average duration;

- the Electronic Dossier is accessible to 50,000 health professionals and 373 pharmacies. There are 5.1 million clinical pictures of patients, 43 million clinical documents are registered, 150,000 visits are conducted daily online. This information, integrated with each other, has enabled better control of treatment interactions and drug administration and increased quality support for professionals' decision-making;
- system development has created a boost for the IT industry in the region: 1,320 IT specialists and 107 companies have been involved at full capacity.

9.3 H2020 ENACT Project Pilot Testing Experience

This framework includes the experimentation conducted in ISRAA (ISTITUTO PER SERVIZI DI RICOVERO E ASSISTENZA AGLI ANZIANI) that is a Public care provider organization for older people, based in Treviso (Italy) concerning the investigation, and subsequent experimentation, of some technological solutions for the remote assistance of fragile people living in the residential context of "Borgo Mazzini Smart Cohousing" foreseen within the Horizon 2020 ENACT DevOps project on DevOps of trustworthy smart IoT systems.

The residential complex in which the elderly people who participated in the pilot reside is located in the historic centre of Treviso. It consists of 46 flats with a total surface area of 5,589 m2 in which elderly people live alone or in pairs with an average age of 75 years.

9.3.1 ENACT Pilot Scenarios on Smart Building and eHealth Impact

As the first step to the e-health IoT system design, in order to understand the attitudes and needs of the elderly residents in Cohousing towards technological innovations for the improvement of quality of life, a set of questions were defined to be asked to the residents in the form of an interview. Then, a focus group followed where eight residents representative of the elderly population used the designed technologies for environmental comfort and independent health management.

Below are some of the most significant elements taken from the survey conducted in August 2020:

The study showed a general inclination towards the adoption of technologies useful for monitoring both one's own health and the living environment for the benefit of one's comfort and safety.

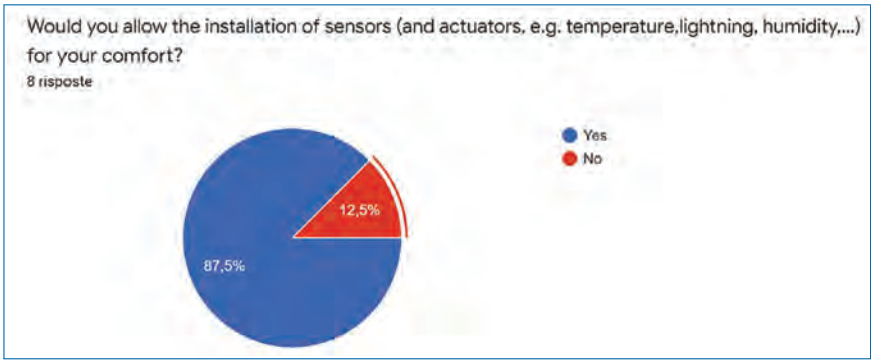


Figure 9.2. Percentage of users who agree to sensors installation for environmental comfort.

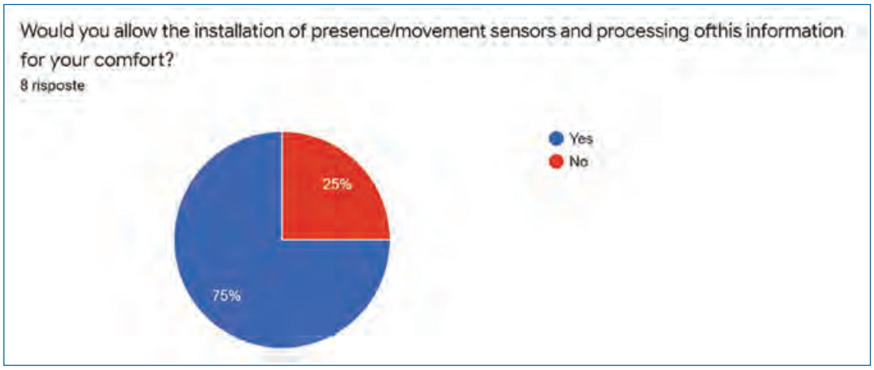


Figure 9.3. Percentage of users who agree to presence and movement sensors.

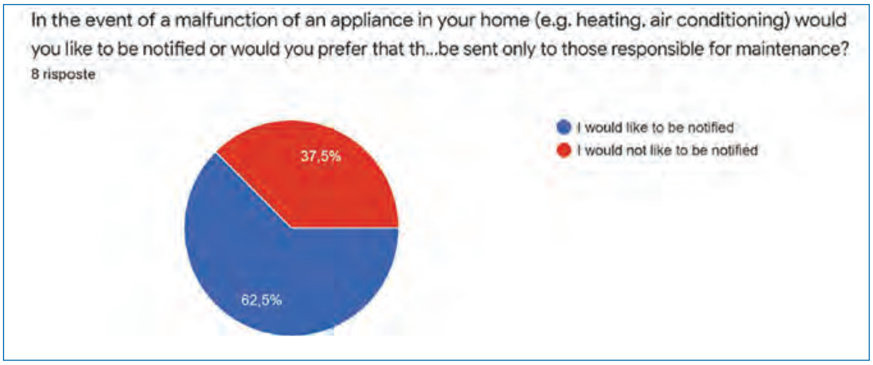


Figure 9.4. Percentage of users who want to get notifications of devices' malfunctioning.

On the basis of this attitude, some residents with conserved cognitive resources and a discreet functional autonomy were involved in order to test the devices provided by the Norwegian Company TellU that provide eHealth solutions in health care such as: thermometer, saturator, Oxymeter, sphygmomanometer capable of

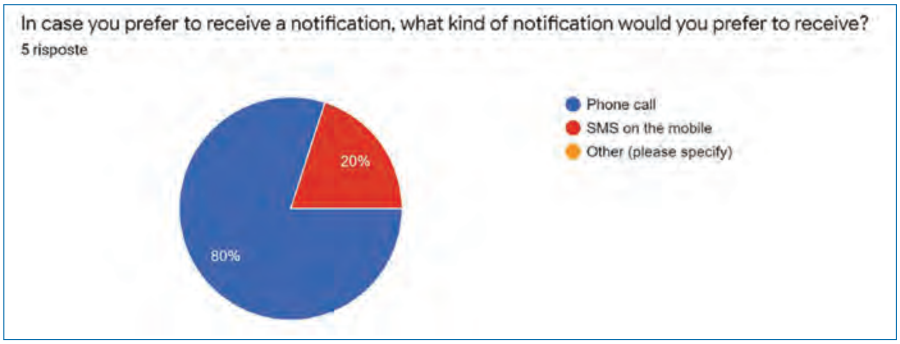


Figure 9.5. Percentage of notification modalities preferred from users in case of malfunctioning.

detecting and transmitting the parameters detected via Bluetooth in real time to the ISRAA care manager. In this way, on the basis of a personalised care plan, innovative teleprotection paths have been activated based on the detection of critical alarm thresholds, for each parameter, over which the care manager was able to act in a timely manner by innovating the care processes.

The experience was favourably in the eyes of the elderly people testing the solution, who were able to experience the benefits of these health support tools, highlighting the high usability of the devices throughout the trial.

With regard to the organisational impact, determined by the experimentation, it should be noted that the nurses and care management staff involved appreciated the time savings, the accuracy of information and the possibility of acting proactively, guaranteeing better health conditions for the people assisted.

9.3.2 Technical Overview of the eHealth Case Study

The industrial-based use case from TellU that was developed in ENACT is a Digital health system for supporting and helping various patients staying at home or in residencies such as cohousing to the extent possible during treatment and care, as well as to have tight interaction with health personnel through digital means in addition to adequate physical meeting points. This makes the patient more independent and it enables support for extensive self care. One type of "patients" supported by the provided digital services is elderly people, for whom the Digital health system will feature elderly care to allow the elderly to live safe at home. Another type of patients are people with chronic diseases such as Diabetes, Kidney diseases, Chronic obstructive pulmonary disease (COPD) and people with temporary diseases such as Covid-19 and cancer. These are patients that need to be regularly followed up and that would benefit from sensor based health status monitoring and digital self care services. For example, Diabetes patients apply sensors and devices to follow

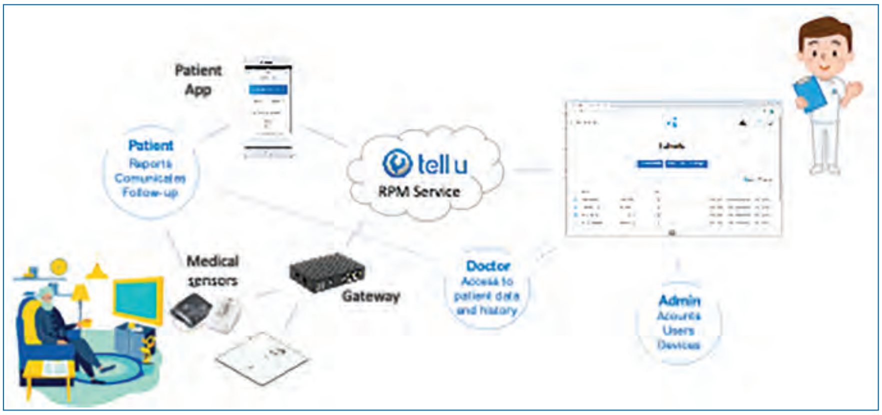


Figure 9.6. The general set up of the ENACT eHealth case study.

their glucose level and regularly provide measurements and questionnaire reports that can be followed up by health personnel. The general set up of the Tellu eHealth case study for medical telecare services is illustrated in Figure 9.6.

The digital health system controls both equipment that are deployed for remote supervision (such as bed sensors, motion sensors, sensors for indoor and out-door location, video based supervision, etc.) and various types of medical devices and specific sensors supporting the care and wellness for the specific patient (e.g., blood pressure meter, sphygmomanometer, Oxymeter, glucose meter, medicine reminder, etc.). In addition, the system can integrate with other systems, for instance to provide information or alarms to response centers, caregivers, physicians, family, etc., and to feed information to medical systems such as electronic health record systems.

In terms of managing the extensive distribution of devices, sensors and software across the IoT and edge space we exploit what we denote "the Personal Health Gateway" (PHG) which integrates the sensors and devices and that controls the edge and ensures the right data are provided to the various stakeholders and to the cloud based system. Thus, the handling of large numbers of largely distributed personal health gateways and their connected sensors has been a main focus in this case study for the validation and exploitation of the ENACT technologies. In particular, we have explored the potential of ENACT for the IoT, edge and cloud services, by having smooth integration of heterogeneous devices, DevOps process for the development of the edge components, as well as secure and trustworthy connections and data transfers. This case study is set up with a local/edge infrastructure consisting of a set of devices and a home gateway (GW). A set of such local infrastructures are then connected and aggregated into a cloud-based infrastructure. The overall technical architecture of the use case including the PHG are depicted in Figure 9.7.

The Personal Health Gateway architecture is the one depicted in the lower part of the figure, and is the element that is controlling the edge and connecting devices

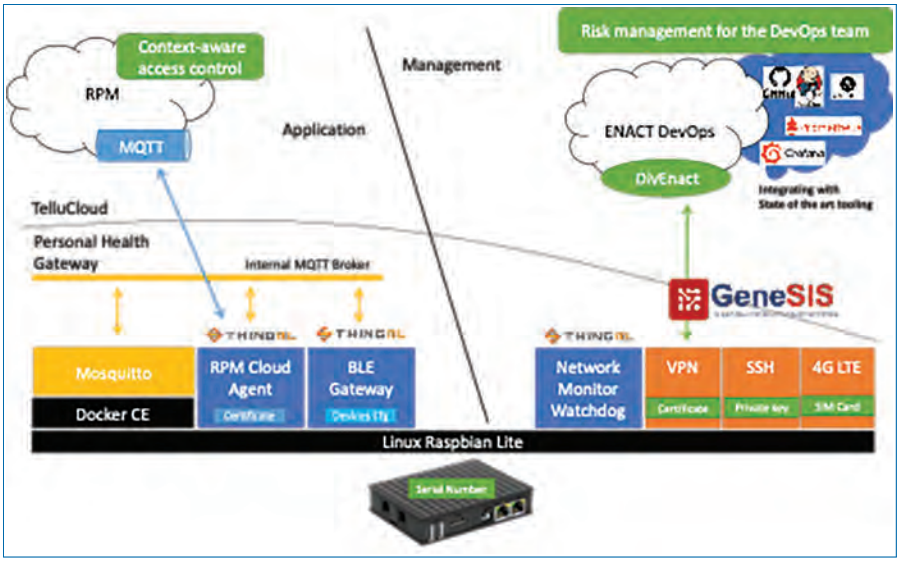


Figure 9.7. Overall architecture with the GW architectural components.

and sensors in the IoT and edge space, while the TelluCloud eHealth system resides in the cloud. The total system encompasses a complex ecosystem spanning IoT, edge and cloud. The Personal Health Gateway consists of a set of microservices to manage the various interactions with the devices and cloud services. The application level interaction and the management is completely separated. This is partly to ensure strict security and privacy requirements. The BLE gateway component manages Bluetooth Low Energy (BLE) enabled devices, for example blood pressure meter, scale, glucose meter, etc. The RPM cloud agent includes the application logic that resides at the edge level and interacts with the cloud level service. A set of microservices supports the management and DevOps process, providing access to system level operations of the Personal Health Gateway through secure channels. Moreover, it includes the monitoring component providing system and application level monitoring required for the continuous operation of the service. The Gateway includes an MQTT broker and support for standard internet communication protocols. The components run on docker containers. The application of the ENACT enablers is indicated in the overall architecture of Figure 9.7:

- The context aware access control is explored to provide more advanced application-level functionality in order to dynamically provide access to different stakeholders based on the context. Context can for example imply an escalated state or a crisis situation. For example, in case of a fire alarm in the patient's house, it may be important to provide further access to the installed camera for example to provide access to firefighters for them to better assess the current situation, while in the normal state the camera will only be possible to be accessed by authorised health personnel;

- ThingML is fully exploited for the efficient coding and DevOps support of the Personal Health Gateway;
- GeneSIS together with DivENACT is explored for the efficient management and continuous deployment of potentially large scale deployments of our telecare service, where large amounts of IoT and edge devices such as welfare sensors and medical devices are managed through the deployed Personal Health Gateways (PHG) residing in people's homes. Note that the PHG is the software stack as depicted in the overall architecture figure above, thus, it may also be deployed on mobile gateways (e.g., smart phones) and we are currently releasing a new version of our PHG that can be deployed on Android and iOS based smart phones, enabling the patient to do medical measurements on travel.
- The ENACT Risk Driven Decision Support tool is explored as part of our DevOps process that needs to be compliant with standards such as ISO 27001, where risk analysis and risk management is required to be an integral part of the DevOps process.

Reference

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

DOI: 10.1561/9781680838251.ch10

Chapter 10

Intelligent Transport System: The Indra Use Case

By Francisco Parrilla, Sergio Jiménez Gómez, Modris Greitans and Janis Judvaitis

10.1 Introduction

The future of the railway market involves digitization, automation, connectivity and the use of intelligent systems that continue to add value to the society by improving management, operation and user experience, in order to face the challenge posed by the European Green Deal[1] as evidenced in the different Strategic and Innovation European agendas,[2] as well as in the plans and reports of public governs and relevant public organizations and Programs (such as the Shift2Rail European Innovation Initiative,[3] the Innovation Plan for Transport and

1. https://ec.europa.eu/info/strategy/priorities-2019-2024/european-green-deal_en
2. https://errac.org/publications/strategic-rail-research-and-innovation-agenda/
3. https://shift2rail.org

Infrastructures launched by the Government of Spain,[4] the French-Swedish Strategic Partnership for innovation, green solutions in the transport sector,[5] etc).

The digitalization of the railway market, as well as the automation and deployment of new intelligent systems, requires the design and implementation of new systems to be deployed in the railway ecosystem; systems that will pose a technological challenge to achieve the objectives set in the European agendas for the coming years, and that will entail major changes as well as the deployment of a large number of devices and subsystems both On Board and On Track. These systems and devices, to face a progressive and changing digitalization of the sector, must be prepared for agile development and deployment, where their control and monitoring provide the necessary mechanisms to guarantee an efficient, robust, safe, secure and updated operation, according to the demands and challenges to be met.

The ENACT results emerges as a facilitator to meet the proposed challenge, in line with the strategic lines and programs described for the future digitalization of the rail market providing DevOps enablers.

10.2 Rationale

The rail domain requires infrastructure and resources that are usually expensive and require a long-time planning and execution. Therefore, the usage of the rail systems must be trustworthy, following strict security and safety regulations. Several functionalities could be implemented within the rail systems to ensure that the system could tackle its high critical requirements as planned.

The proposed Use Case for railways shows how the use of the ENACT enablers can be used to enhance the DevOps cycle of new innovative systems – aligned with other innovation programs mentioned in the Chapter 10.1 – exploiting and evaluating their potential. The selected innovative systems have been analyzed, implemented and tested:

- **On Board WTI (Wireless Train Integrity)**[6]: This functionality is in charge of measuring, in real time, train composition parameters and evaluate them

4. Ministerio de Transportes, Movilidad y Agenda Urbana. Gobierno de España: “Plan de Innovación para el Transporte y las Infraestructuras”. February 2018

5. https://www.tresor.economie.gouv.fr/Articles/2018/03/28/partenariat-franco-suedois-pour-l-innovation-et-les-solutions-vertes-french-swedish-partnership-for-innovation-and-green-solutions

6. Aligned with On Board Train Integrity Technology demonstrator tasks defined on TD2.5 addressed on X2RAIL-2 and X2RAIL-4 projects on which Indra is involved https://projects.shift2rail.org/s2r_ip_TD_D.aspx?ip=2&td=061d0fcf-51a6-4358-a74f-a4d34e8dac01 and making use of SCOTT https://scottproject.eu/ and DEWI results http://www.dewiproject.eu/

to report the train integrity status. The On Board system, based on WSN Sensors among the composition, provides the necessary information to determine the rolling stock material that composes the consists and evaluate and ensure, through an On Board unit, its integrity status. This integrity status is shown to the driver through the Train Management Systems (TMS) and a Cloud service interface.

- **Logistic and Maintenance System**[7]: This functionality provides information to register and locate both rolling stock material and on-track signalling devices and inform about their status. This functionality is required to solve the rail environment needs to locate and monitor the status of the big heterogeneity and flexibility of the compositions and signalling devices, making a special emphasis on the freight compositions, to optimize the rail business operation. The points that are optimized into the rail framework are the management of the rolling stock, cargo tracking, etc. To this end, it is required deploying IoT On Board and On Track, together with Cloud solutions to track and manage the rolling stock material data and to perform predictive maintenance.

These services are illustrated in the following Figure 10.1:

The DevOps role in the Use Case assists to reach this automation and digitalization objective, with the following focuses:

- **Security and Privacy Monitoring (S&P Mon&Con) tool**: This tool is responsible for monitoring and actuating over the On Board infrastructure to guarantee its security characteristics.
- **GeneSIS tool**: This tool monitor performs software remote deployments on the rail equipment to keep all the devices with the desired software version.
- **Behaviour Drift Analysis (BDA) tool**: This tool monitors the behaviour of the equipment to detect deviations related with the proper define behaviour for them.
- **Actuation Conflict Management (ACM) tool**: This tool detects conflicts that may appear in the Use Case operation and to help to resolve the conflicting actions.
- **Testing and Simulation tool**: It is a tool that simulates a part of the Use Case infrastructure and makes a Digital twin of it to be evaluated.

7. Aligned with Smart radio-connected all-in-all wayside objects demonstrator tasks defined on TD2.10 addressed on X2RAIL-1 and X2RAIL-4 projects on which Indra is involved https://projects.shift2rail.org/s2r_ip_TD_D.aspx?ip=2&td=061d0fcf-51a6-4358-a74f-a4d34e8dac01 and making use of SCOTT https://scottproject.eu/ and DEWI results http://www.dewiproject.eu/

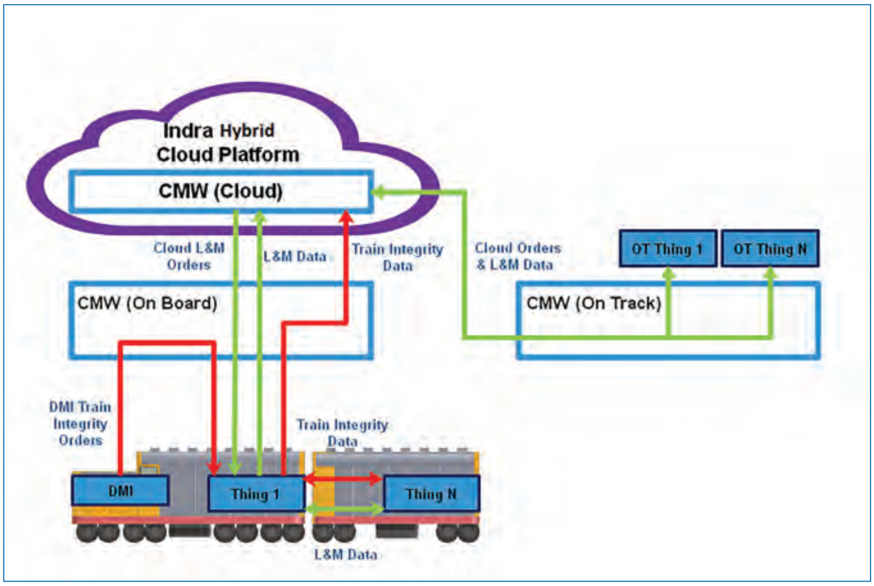


Figure 10.1. On Board WTI (Wireless Train Integrity & Logistic and Maintenance System Functionalities Scheme).
Source: INDRA.

- **Root Cause Analysis (RCA) tool**: This tool detects possible failures that may occur in the Use Case infrastructure informing about the most likely cause for that fail.

One of the challenges that the Use Case faces to reach the automation and digitalization of the rail environment is the scalability. The mentioned DevOps tools developed in ENACT for the Use Case have different objectives. However, these objectives converge on solving the scalability issues that the functionalities hide. To evaluate the scalability impact and to provide the issue's magnitude, a real rail scenario example is provided. An example to show the scalability issues is the Madrid-Barcelona French Border line, one of the first high speed lines built in Spain and one with the higher capacity (more trains per day). An example of a real line magnitude can be seen in Figure 10.2.

For this real example, focusing only on On Board systems, there are 94 trips circulating through this line: 47 of them from Madrid to Barcelona and the rest from Barcelona to Madrid. Each trip is accomplished by a single train composition that could be built following three different kinds of composition: simple, double, and mixed.

- **Simple**: There are 38 simple train compositions per day where there is only one locomotive wagon. Each simple composition is formed by 12 wagons.

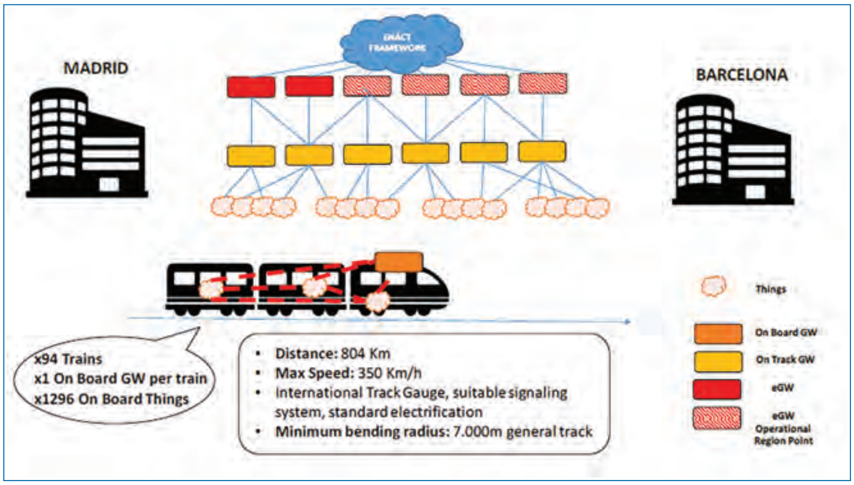


Figure 10.2. Madrid-Barcelona French Border rail line general characteristics. *Source*: INDRA.

- **Double**: There are 5 double compositions per day where there are two simple compositions joined. Each double composition is formed by 24 wagons.
- **Mixed**: There are 5 mixed compositions per day which are formed by different kind of pieces of rolling stock. Each mixed train composition is formed by an average of 18 wagons.

In the real line, there are running 1296 wagons in average, if both directions of the line are considered (94 trains per day). Based on the functionalities architecture, further details shown in Section 10.3, it is estimated that around 4000 units of On Board equipment are required to cover the On Board equipment for this line.

From the magnitude presented, we can verify that it is essential to provide a deployment mechanism to update all these systems in a controlled, automated and orchestrated way. It is important to remark that these systems should be developed with digital twins in mind to test the potential impact of new updates in a test environment. They faithfully reproduce the potential impacts of these updates in operating environment with mechanisms that ensure the cybersecurity of communications throughout the ecosystem and adjusts the deviations of the sensor networks involved. This serves to ensure the robustness of the different orders distributed throughout the global system.

10.3 Use Case Implementation

In this section, we explain the IoT platform brought to implement the rail functionalities and show how the ENACT enablers are applied. Moreover, it includes how

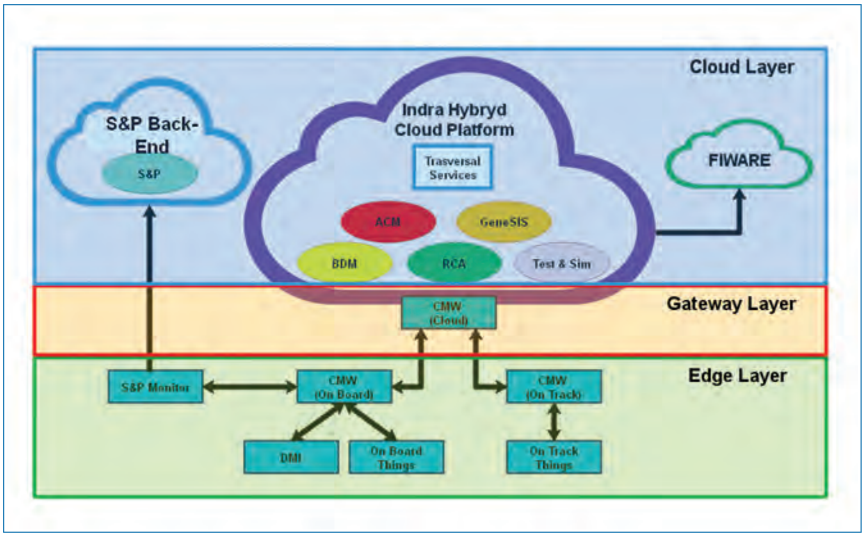


Figure 10.3. Rail Use Case architecture enriched with the DevOps tools. *Source*: INDRA.

the DevOps tools enhance the Use Case itself. As illustrated in the Figure 10.3, we identify three different architecture layers. These layers are the Edge, the Gateway, and the Cloud layers:

10.3.1 Edge

The edge layer is made up of the various objects (Wireless Sensor and actuator networks – WSAN – and other concentrating and/or communication devices) distributed both on the track and in the On Board equipment. This layer also serves to provide/send raw data to the functionalities. It is used by the BDA (10.4.5) and ACM (10.4.4) tools to perform conflict analysis and monitor the behavior of the devices Moreover, it is used by the Testing and Simulation tool to create a fair Things Digital Twin. The following elements form it:

- **Things**: The key element in this layer is defined under the name of Thing. The Things provide the functionality parameters described above.
  The Things are defined as a group of nodes which globes coordinators, sensors or actuators. In such a way, the Things have several capabilities: gather, actuate and communicate.

  It must be emphasized that the layer covers the On Board and On Track sections. WTI and a set to Logistic and Maintenance parameters are obtained and processed in the On Board section, while the On Track section participates in the provision of another set of Logistics and Maintenance

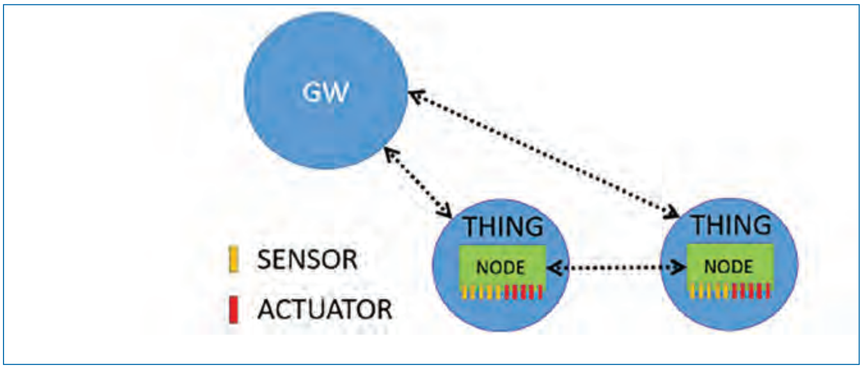


Figure 10.4. On Board and On Track Things.

Source: INDRA.

specific data. The On Track infrastructure uses RFID technology to obtain information from the train. These Things are powered up by an energy harvesting system to cover the case that no electrification systems are equipped On Board a train.

- Sensors:
  - On Board: an accelerometer, a Received Signal Strength Indicator (RSSI) sensor, and a Global Navigation Satellite System (GNSS) receiver, and Radio Frequency Identification (RFID) tag and reader.
  - On Track: RFID reader.
- Actuators:
  - On Board: Light-Emitting Diode (LED)s and displays.

The Things software consists of 5 modules: data managing module, inauguration module, integrity module, logistics module and maintenance module. They are working in synergy to provide the data and control over the On Board and On Track Things infrastructure deployed on the rolling stock.

- Data managing module:
  - *On Board*: The data is gathered at the nodes located on each wagon of the train. Data are sent over the air using the IEEE 802.15.4 based ZigBee protocol for secure and reliable communications. As typical for wireless sensor and actuator network, there is at least one so called base station, which forwards the data to the next processing point. The system is also capable of communication in the other direction sending commands from base station to the nodes.

    In this case the next processing point is WSN coordinator. WSN coordinator acts as a data forwarder from Things to the Communication middleware (CMW), while also ensuring that the data forwarded

  are in the correct format. WSN coordinator is also responsible about the forwarding of commands to the Things from CMW. The commands from CMW can be: start inauguration, start/stop integrity control, start/stop logistics control, start/stop maintenance control and reset. All of the commands are part of modules described in next sections.
- *On-Track*: The on-track data gathering does not require the wireless network, the RFID sensor readings are gathered and forwarded to the CMW. On-track infrastructure listens to the following commands from CMW: start/stop logistics control and reset.

– Inauguration module: The inauguration module is implemented on the WSN coordinator. After receiving the command about the start of inauguration procedure from the CMW, the base station is notified to identify the nodes located on the wagons belonging to this train and check that they are operational. In this module also the physical order of the wagons is calculated using the GNSS and RSSI sensor values and RFID data. After successful inauguration the WSN coordinator sends corresponding message to CMW reporting the inauguration status.

– Integrity module: The integrity module is operated only in the On Board infrastructure and is responsible for continuously validating the train integrity while it is operational. Integrity control can only be started after a successful inauguration phase is finished and integrity control is not already operational. When the WSN coordinator receives the command from CMW to start the integrity control it notifies the base station to send out the integrity start command. After start command each of the nodes and base station reports sensor status every 250 ms or 4 times per second. The received sensor data is forwarded to the CMW as raw data consisting of GNSS position, accelerometer data and RSSI value measured at the base station for each wagon. Using this information also the train integrity is calculated, it is based on aforementioned sensors and train integrity is considered lost in case when at least two of three sensors report data that indicates that there is a train integrity issue. The train integrity information is also forwarded to the CMW. The train integrity system remains operational until the stop or reset command is received. If the start inauguration or start integrity command is received while the integrity system is operational, the command is ignored.

– Logistics module: The logistics module is responsible for providing the business information about the train logistics. Logistics module operates on the On Board and on-track infrastructure and uses the RFID reader data. On-track infrastructure provides wagon order information, but On

Board infrastructure provides information about the cargo and also allows to identify the wagon by RFID tag associated with it. The logistics module can be started after successful inauguration and stopped at will.

- Maintenance module: The maintenance module is responsible for providing the maintenance information about the wagons and infrastructure. While the maintenance module is operational it reports the node energy consumption and battery charge status. The maintenance module can be started after successful inauguration and stopped at will. Also the maintenance module handles the reset command, which can be issued by CMW and will reset the Things software in case of any errors. The reset command will not be accepted by the WSN coordinator if inauguration process is running or integrity control module is operational.
- Testing and validation: The developed infrastructure was validated using the demo setup of Lego train with minimal changes to the setup described above due to physical limitations of Lego train. The demo setup with Lego train was used to provide the partners with sensor, logistics and maintenance data from the on-track and On Board infrastructure while located in the lab environment but still providing non-generated data, thus making it easier to test, integrate, debug, and showcase developed technologies.

- **DMI**: The DMI (Driver Machine Interface) is included into this layer and it is responsible for managing the Things into the train by the driver and to perform Safety data treatment for the functionalities.
- **S&P Monitor**: This equipment is the key element in the security properties for the Use Case Edge layer. The S&P Monitor is an architecture component located On Board in charge of monitoring the traffic that is carried by the gateway layer 10.3.2. It must be emphasized that the traffic analysis ignores the business data related with the ITS functionalities. The reason behind this decision is that the functionality is in charge of covering the security aspects of the On Board equipment out of functionality focus. This elements is formed by several differentiated entities:
  - Hardware: The hardware offers the computing capacity to the software, the connectivity with the gateway by Ethernet and the connectivity with the S&P Mon&Con Back-end through a 3G/4G interface.
  - Monitor SW: The monitor software sniffs the traffic from the gateway in order to be treated and to be sent to the S&P Mon&Con Back-end for a further analysis.
  - User's removal Software: The S&P Mon&Con Back-end analysis could throw as a result that the users that generates the traffic monitored are intruders, therefore, a notification with the users is published to this

entity. This entity, developed by Indra, revokes the user in the ITS central authentication services.

10.3.2 Gateway

The gateway, so called CMW (Communication Middleware), is the element that connects all the Use Case system elements. It gathers the edge data and provides it to the Cloud. Moreover, it gathers the orders to the Things to start the functionality services. This CMW is based on Indra developed solution certified for rail environments.

From a general perspective, the basic functionalities of this CMW are as follows:

- Routing the messages from the different services to be sent to the different architecture entities. This routing follows a preconfigured topology to guarantee the provision of the information in a safe and secured manner.
- Synchronization tasks to keeps all the devices working with the same time reference.
- Authentication tasks at different OSI levels to enable the connectivity of the edge elements and to enable the connection of the CMW with the Cloud layer.
- Provision of CMW status metadata to be evaluated by the DevOps tools.

The Use Case, into the ENACT project framework, relays part the scalability issues to the CMW. As the CMW equipment cost is high to provide several of these devices to the project, considering the budget, the scalability is tests will be done in a single physical CMW and several virtual ones will be provided for testing. Therefore, the general architecture is formed by a single physical CMW located On Board and the several virtual CMWs.

10.3.3 Cloud

The Cloud layer is formed by the Cloud platforms that participate on the Use Case and how they are related. Three different Cloud platforms are considered: FIWARE [1], Indra Hybrid Cloud Platform, and S&P Back-end.

- **Indra Hybrid Cloud Platform**[8]: The cloud is a hybrid solution that combines a private and public part. The private par is in charge of the internal

8. https://www.igi-global.com/chapter/security-in-rail-iot-systems/258896

ITS task such as edge data storage, edge elements authentication, and external partner's authentication. The public part is in charge of integrating external elements such as tools or other Cloud platforms. This is the integration point made for all the Cloud resources and DevOps tools that require it. All the DevOps tool, except the S&P tool, are integrated in this layer to accomplish their functions. Moreover, it is the integration point also for the FIWARE Cloud platform.

- **FIWARE**: FIWARE is an open source platform that the European Union supports as a future platform to provide Cloud services. For this Use Case, the FIWARE tool can allocate several functionalities that goes from systems authentication, storage, DevOps tools integration. However, for the ENACT project the functionalities uses this Cloud platform to provide to the Use Case user several dashboard to track the Use Case functionalities (making use of the FIWARE ORION component for integration tasks and FIWARE Grafana component as the presentation tool).
- **S&P Mon&Con Back-end**: It is a service in a separate Cloud that supports the security monitoring capabilities. The traffic and security data treated in the S&P Monitoring tool is sent to this back-end to be evaluated. The service provides to the Use Case user with an interface to interact with the traffic behaviour and security rules that are desired for the On Board installation, as well as it offers the situational awareness functionalities for the user to get informed on the security status at all times.

10.4 DevOps of ITS System Powered by ENACT Tools

This section is intended to introduce the specific functionalities that the DevOps tools use and how these has been implemented and tested to enhance the Use Case itself.

10.4.1 Security Monitoring (S&P Mon&Con)

The Security and Privacy Monitoring and Control Enabler (for short, S&P Mon&Con) is in charge of providing the security layer that it is required for the Edge, as IoT technologies can suffer multiple types of attacks at this layer. The monitoring service of the tool is the one used in the Use Case. The S&P Mon&Con is the only tool that it not integrated into the Use Case at the Indra Hybrid Cloud Platform, as it acts as supervisor of the Use Case IoT system checking from outside whether resources at the Edge could be under compromise, thus, it has a parallel

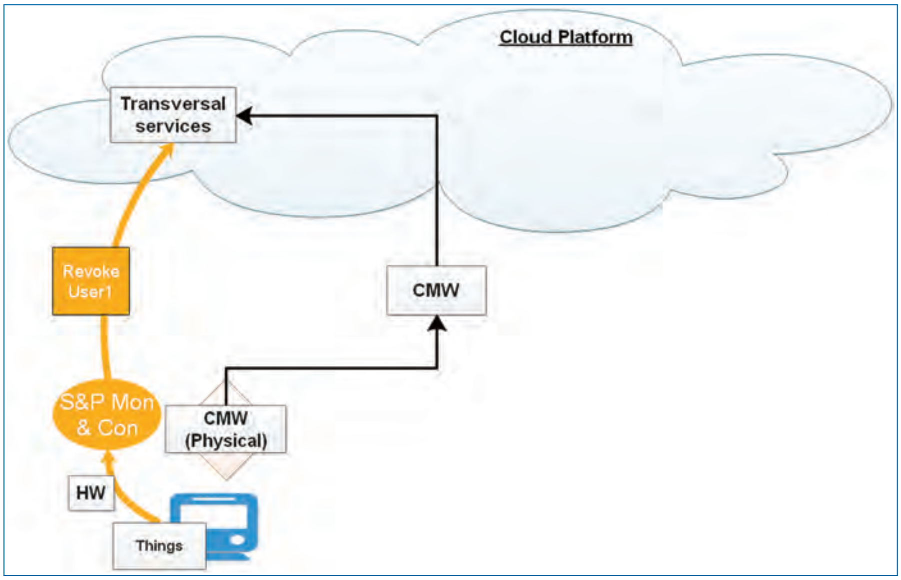


Figure 10.5. S&P – Rail Use Case integration synoptic.
Source: INDRA.

deployment and a different integration philosophy. A synoptic of the scenario can be seen in the Figure 10.5.

The tool is divided into two sections, the monitoring and the actuation part. The monitoring part is in charge of collecting the data that are going through the On Board gateways, this data includes the Things and the DMI data only. These data is monitored and sent throw a specific On Board device to the S&P Monitoring Back-end to be analysed. From this analysis the following parameters are analysed:

- **<u>Traffic behaviour monitoring</u>**: The traffic behaviour is considered as one of the characteristics that defines the rail functionalities is regular. The Safety and Security requirements that defines this kind of functionalities require a regular traffic that may be affected by intruders. The S&P tool is focus on detecting deviations in this edge layer traffic and revoking the user, from the Use Case central authentication servers, that generates that deviations.
- **<u>Intrusion detection</u>**: The authorised users are those systems that can publish/subscribe to IoT Platform (authenticated in the central Use Case authentication server), and are registered in the S&P Monitoring tool registry (both user id and Media Access Control (MAC) addresses). In case a user is authenticated in the central Use Case authentication server but not in the S&P Monitoring tool database, that user or/and the device they use will be revoked

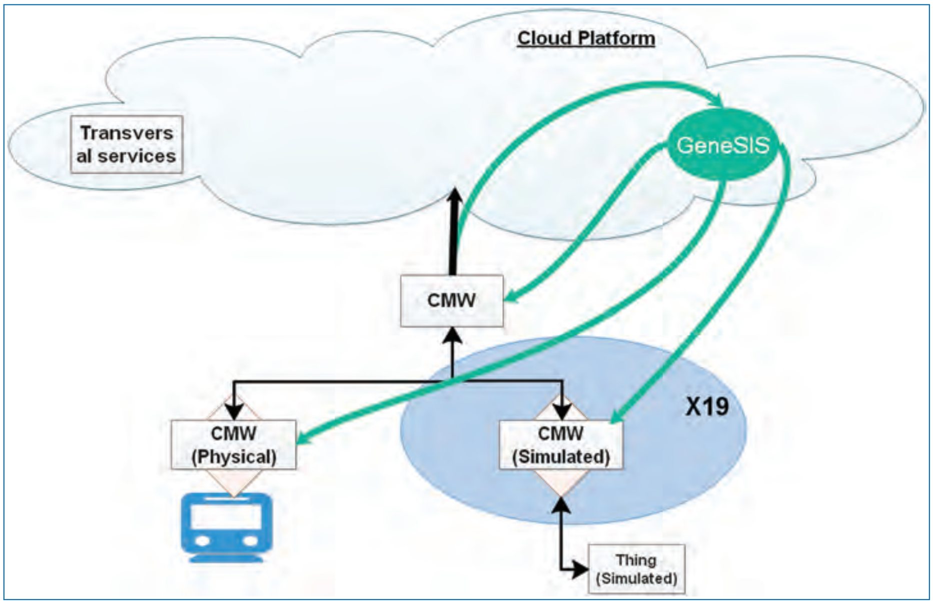


Figure 10.6. GeneSIS – Rail Use Case integration synoptic.
Source: INDRA.

in the Use Case central authentication servers, banning its connectivity to the whole IoT Platform.

10.4.2 Automatic Deployment – GeneSIS

The tool is in charge of managing and controlling the software that is running in the ITS Use Case.

It deploys the Docker images running in the gateways and ensures that they are correct and deployed with the access credentials requires to be integrated into the ITS infrastructure. A synoptic of the scenario can be seen in the Figure 10.6.

The tool also is aware of the status of the system deployed. The tool monitors the status of the Use Case infrastructure to check if it is possible making a deployment or not. In case the system is running the Use Case functionalities, the deployment cannot be performed. Moreover, the tool is able to evaluate if the deployment is correctly performed and if it has a conflict with previous deployments made in the same device.

10.4.3 Testing and Simulation

The testing and simulation tools has two roles into the ITS Use Case. The first one is monitoring the infrastructure in order to validate that the ITS infrastructure

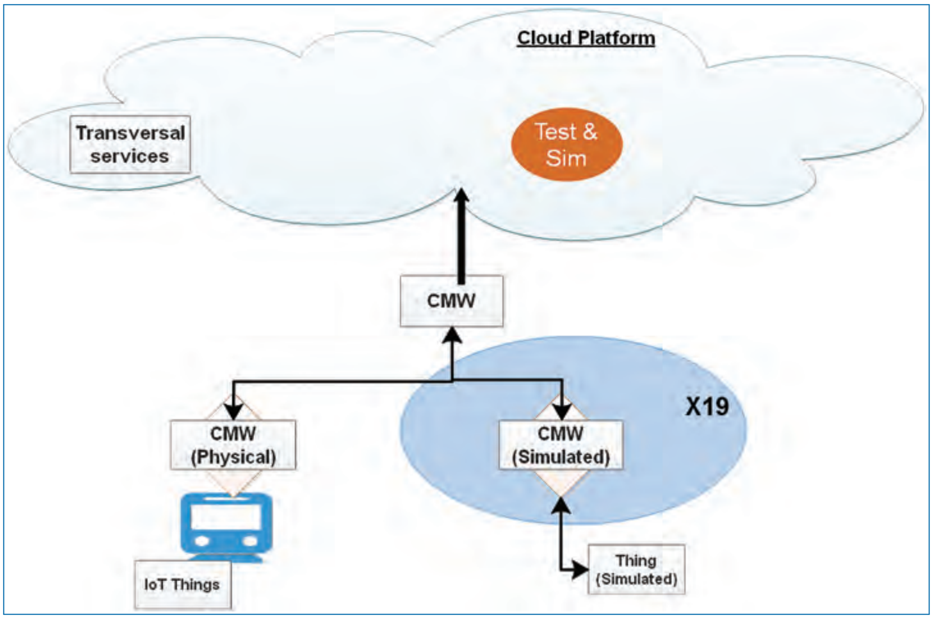


Figure 10.7. Testing and Simulation – Rail Use Case integration synoptic. *Source*: INDRA.

is running properly and to simulate failures into the infrastructure to enhance the validation value against issues that may appear into the operation of the Use Case itself. A synoptic of the scenario can be seen in the Figure 10.7.

The Testing and Simulation tool collects all the data gathered by the gateway layer to monitor its workload when it is operating the rail functionalities' tasks. In this case, all the traffic from the Edge and Cloud layers is monitored, in other words, the 100% of the traffic managed during an operation by the gateway.

Using these data the tool is able to replicate a single gateway, in a simulated virtual environment, taking as a basis the behaviour of the monitored gateway. This procedure that replicates the devices in a virtual infrastructure is called Digital Twin. Generating several Digital Twins as many times as desired provides the tool's user the possibility to generate a virtual scenario with all the gateways desired; hence, the scalability of a scenario can be proved. Moreover, as the virtual infrastructure is generated, the tool is able to simulate certain situations that may compromise the infrastructure and check its reliability.

This simulated environment is not enough to test the system's scalability. Several metrics to evaluate the simulated gateways are required. A specific report about the gateways status is required to know how the gateway is dealing with that monitored data. As it is mentioned in the Section 10.3.2, the gateways report specific data about its physical and routing status to evaluate their operation. Therefore, the tool is able to infer, using these reports, the behavior of this simulated infrastructure in any situation proved.

10.4.4 Actuation Conflict Management (ACM)

The ACM tool is intended to manage conflicts that may appear in the train operation. This tool is based on generating behavioural models that can be applied to the On Board devices to fix an specific behaviour depending on the rail system inputs. A synoptic of the scenario can be seen in the Figure 10.8.

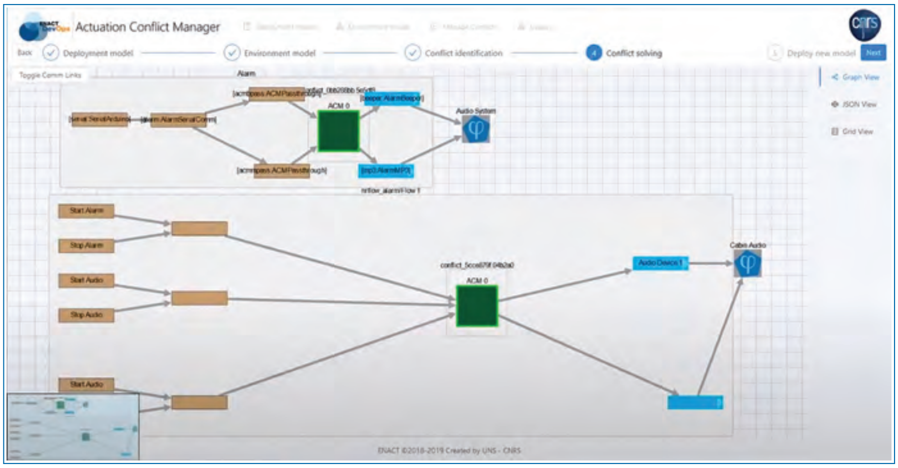


Figure 10.8. ACM – Rail Use Case integration synoptic.
Source: INDRA.

Based on the operation iterations, the developer can check the conflicts that may appear. The Safety and Secure functionalities defined are designed to not generate conflicts in the operation; however, the non-Safety systems that uses the safety ones may generate conflicts between their behaviours (e.g., audio announcements, alarms, etc.).

The ACM tool is able to generate models, as shown in the Figure 10.8, which can be deployed in any device that is the root cause of the conflict.

10.4.5 Behavioral Drift Analysis (BDA)

The BDA in charge of checking that the Things behaviour matches with the real modelled behaviour defined for them. The tool monitors the business data published by the Things to the Cloud layer to check in real time deviations with the mentioned model.

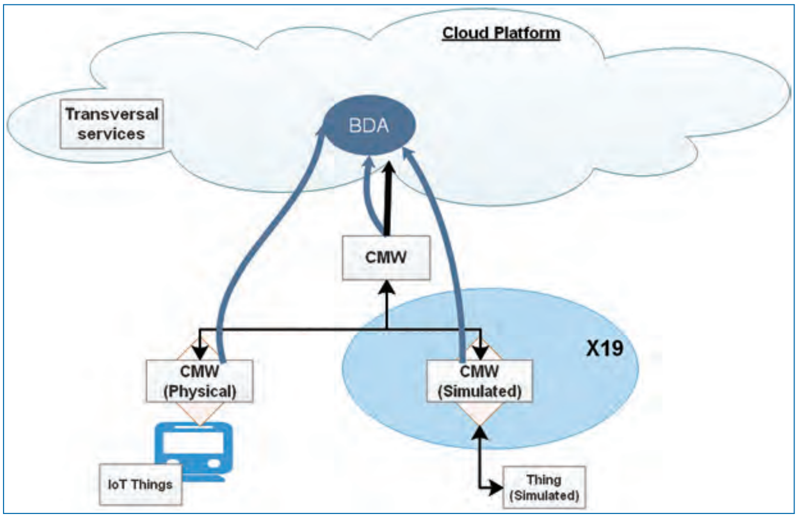


Figure 10.9. BDA – Rail Use Case integration synoptic.
Source: INDRA.

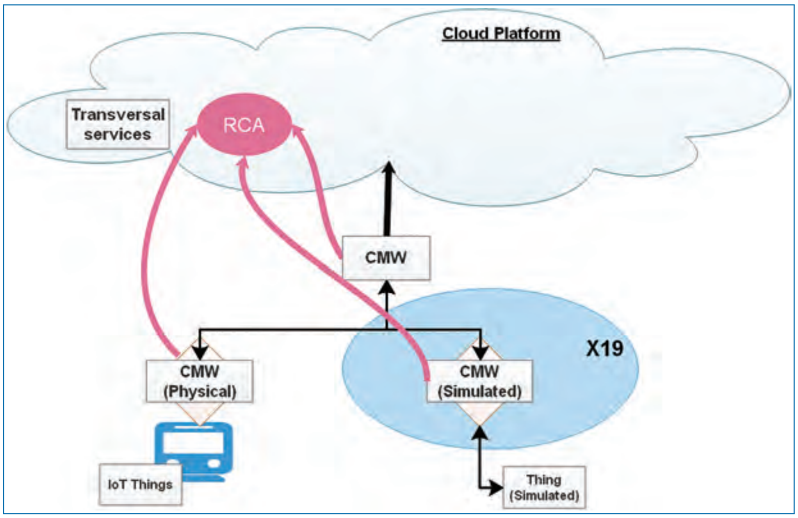


Figure 10.10. RCA – Rail Use Case integration synoptic.
Source: INDRA.

10.4.6 Root Cause Analysis (RCA)

The RCA tool is in charge of checking the security perspective of the ITS infrastructure from the gateway to the cloud layer finding the reasons for a failure that may occur (Figure 10.10). This tool serves to identify the cause of network mistakes that in a scalable systems is critical due to the quantity of devices deployed.

Several failure scenarios are simulated and recorded by the RCA tool in order to store the behaviour of the infrastructure in case of those failures appear. The data monitored is the physical and routing data reported by the gateways.

10.5 Conclusion

We have tested and validated – focused on developing trustworthy and ready-to-use scenarios to test the DevOps Enablers – the ENACT DevOps framework. From the Use Case perspective, the provision of an IoT platform with the characteristics exposed along the chapter makes a first step to enhance the rail environment and improves the new systems for the future digitalization and automation to an industry that is experiencing a huge evolution in the last decades.

The main impacts that the DevOps philosophy have on the rail environment, based on the mentioned DevOps tools that are listed along the document, are focus on:

- Ensuring the scalability of the system before deploying it. Introducing a Testing and Simulation tool permits the optimization of the infrastructure components and reduces the uncertainty about the system reliability. Therefore, the on-site testing time is reduced.
- Fast and agile deployment of new software versions in a remote manner, this reduces the infrastructure functionalities updated and the maintenance costs.
- Provision of a security backup to the internal rail authentication mechanisms covering since the devices to the application layers. The S&P tool locates the system intruders in a more accurate manner and the by design rail Security and Safety aspects are increased.
- Monitor the deviation in the behaviour of this systems in real time during the operation. The BDA tool helps to locate devices issues in a more accurate manner and, then, as the diagnosis time is improved the maintenance time and tasks is reduced.
- Ensure the interaction between the drivers and the rail functionalities to reduce the human errors. The main impact is the reduction of the failures that may occur in the system caused by the driver. This is highly relevant as the driver is a key factor to rely on the security and safety aspects in the rail environment.

Reference

[1] Indra Sistemas S.A. Contribution. *X2Rail-1 Deliverable D7.2 – Railway requirements and Standards application conditions – Indra Sistemas S.A Contribution*. Tech. rep. May 2018.

DOI: 10.1561/9781680838251.ch11

Chapter 11

Smart Building: The Tecnalia KUBIK Use Case

By Miguel Ángel Antón, Rubén Mulero, Sheila Puente, Larraitz Aranburu and Sarah Noyé

11.1 Introduction

Buildings have long been equipped with sensors and actuators to automate their control. Smart buildings are those whose facilities and systems (air conditioning, heating, lighting, access control systems, etc.) allow integrated and automated building management and control to increase energy efficiency, security, and usability. With the democratization of the Internet of Things (IoT), the number of sensors and actuators is constantly increasing, giving ways to new applications. The reduction of sensors and actuators cost is driving a digital shift in the building sector.

The need for better energy resource control and the requirement to provide better comfort for the user has led to a new market of complex Smart IoT Systems able to provide a vast array of new services or applications to the end-user. Extending legacy system to take advantage of those new services can be expensive, thus limiting possibilities. There is a need for a seamless way to integrate solutions from different manufacturers as well as to ensure effective design, deployment, and operation of simultaneous IoT applications that respect security and privacy requirements.

In that sense, software needs to be changed when new IoT devices are added or new functionalities for user comfort are developed. Therefore, it is necessary to ensure the continuous development and update of the IoT applications. The thermal and climate control needs to be continuously adapted to the environmental changes to keep the occupants' comfort in the buildings. At the same time, potential conflicts between IoT applications acting on the same actuator or the same physical parameters must be identified to guarantee the buildings' trustworthiness. And finally, cybersecurity threats need to be identified and mitigated to preserve the security and data privacy.

By leveraging the ENACT DevOps framework, secure and trustworthy IoT applications can be developed based on the interoperability and orchestrated operation of multiple sensors and actuators.

The smart building KUBIK, situated close to Bilbao, Spain, was inaugurated in 2010 as an experimental infrastructure for developing and validating innovative products and systems to optimize energy efficiency in buildings [1]. It is a three floors building owned by Tecnalia and designed for testing and research ranging from passive systems such as modular insulating components for roofs and facades, to energy generation based on renewable energy and climate control systems. It includes more than 700 sensors and actuators, central Building Management System (BMS), local Renewable Energy Systems RES (RES), local weather station, and Combined Heat and Power (CHP) equipment on-site. In the context of the ENACT project, KUBIK provides the required equipment and well-known boundary conditions for the testing and validation of the enablers developed in the project.

KUBIK experimental infrastructure is relevant to the ENACT project due to its special needs and characteristics such as the combination of legacy building automation systems and new smart IoT devices, this fact requires an interoperability platform to communicate both systems. At KUBIK, several energy efficiency applications and user comfort applications share common actuators (fancoils, lights, blinds, controlled sockets, etc.) which generate actuation control conflicts. Thermal control of a building is also a trade-off between energy consumption and user comfort that must be adjusted to the specific physical characteristics of the building and user preferences. Behavioral drifts in the control of building systems also need to be identified and addressed. And finally, security and privacy of the communications is a must, paying special attention in secure actuation.

The ENACT enablers in combination with the SMOOL middleware platform have been used to solve the challenges described in the previous paragraph. Now, IoT applications are designed, developed and improved using the DevOps strategy, as ENACT enablers ensure no actuation conflicts, security (secure communications, access control, threat detection, etc.) and trustworthiness (self-learning controls, behavioural drifts identification, etc.) saving time and effort.

As a general result of the ENACT project, KUBIK building was leverage to become a place 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. Now, the KUBIK building offers an environment where new IoT devices can be introduced and a wide range of IoT applications for smart buildings can be deployed and tested.

Section 11.2 describes the KUBIK building that was used as an experimental platform to validate the ENACT enablers for smart buildings. Section 11.3 gets into the detail of the technical architecture of the Smart IoT System of the KUBIK building. In Section 11.4, we expose different test scenarios and the benefit of the ENACT enablers. Finally, Section 11.5 concludes.

11.2 The KUBIK Smart Building

KUBIK is an experimental infrastructure focused on the development of new products and systems that provide energy consumption reduction for the building and increase user comfort (Figure 11.1). Its uniqueness lies in its ability to generate realistic scenarios to test energy efficiency resulting from the integration of constructive solutions, air conditioning and lighting systems, and energy supply from conventional and renewable energies. The building contains three floors with different testing zones and a cellar. Its ground floor is an apartment. It has a bedroom, a kitchen, a living room, and a corridor where engineers can test the Energy Efficient Building scenarios for a real home.

The ground floor has various IoT devices installed. Figure 11.2 shows the sensors and actuators installed at the ground floor. There is a flood sensor, sensors on doors

Figure 11.1. KUBIK by Tecnalia.

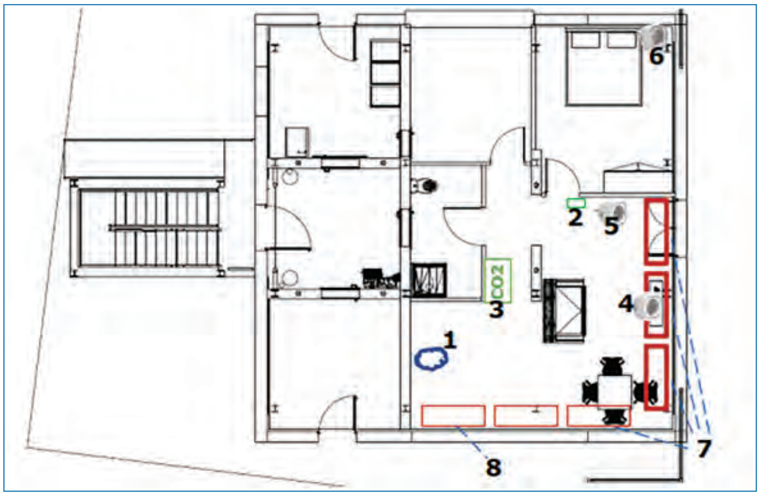


Figure 11.2. Floor plant of the Ground Floor of KUBIK.

Table 11.1. Device/Signals on the Ground Floor and Floor Plant of KUBIK Building. (K) values represents Kitchen, (B) values represents Bedroom, (L) value represents Living Room, (C) value represents corridor.

ID	Type	Device type	Location	System	Signal
1	Sensor	Water Flood	K	Z-Wave	Flood alarm state: ON/OFF
2	Sensor	Door Multisensor	B	Z-Wave	Position: OPEN/CLOSED
3	Sensor	CO2 Sensor 1	K/L/C	Z-Wave	CO2 level 0: 0 ppm to 200 ppm
4	Actuator	Remote Socket 1	L	Z-Wave	Switch state: ON/OFF
4	Sensor	Remote Socket 1	L	Z-Wave	Energy consumption: Watts
5	Actuator	Remote Socket 2	L	Z-Wave	Switch state: ON/OFF
5	Sensor	Remote Socket 2	L	Z-Wave	Energy consumption: Watts
6	Actuator	Remote Socket 3	B	Z-Wave	State: ON/OFF
6	Sensor	Remote Socket 3	L	Z-Wave	Energy consumption: Watts
7	Actuator	4 Blinds motors	L	PLC	Position: UP/Down
8	Actuator	2 Blinds motors	K	PLC	Position: UP/Down

that indicate open or closed status, various electrical sockets sensors and actuators, and motors for the blinds. Except for the blind motors, the devices are wireless sensors and actuators integrated as an additional layer to the building control system. Table 11.1 shows a detailed description of each device represented in the floor plan of the ground floor of KUBIK, its location in a specific room, its belonging to the IoT Smart Space or the wired Building Control group, and finally, the measures or commands it provides.

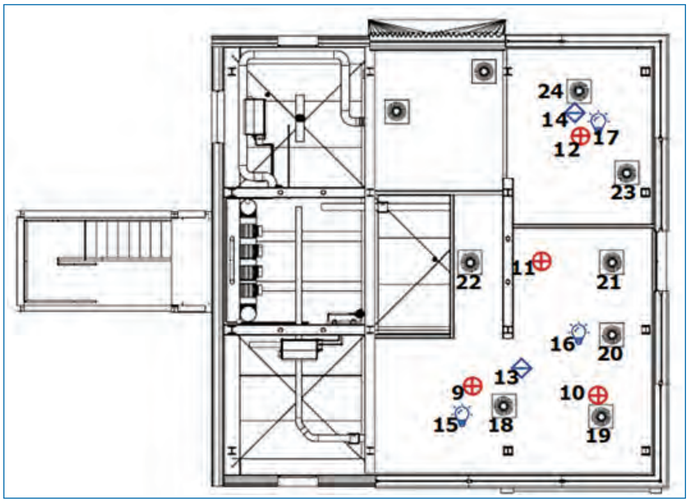


Figure 11.3. Reflected Ceiling plant of the Ground Floor of KUBIK.

Similarly, the reflected ceiling plan of the ground floor of KUBIK with sensors and actuators in their approximate location is shown in Figure 11.3. Those sensors are wireless Z-wave sensors monitoring ambient conditions (temperature, humidity, lighting, and occupancy) and smoke detectors. Sensors and actuators of the fan coil units of the space are connected to the building wired control system. Table 11.2 gives the details of different sensors and their signal types.

The cohabitation between hard-wired legacy sensors and easy-to-install additional wireless sensors presents an interesting case in line with the desire for flexibility and evolution of the smart building applications in the market.

11.3 Technical Architecture

A set of sensors, actuators, and devices are deployed inside each floor of the KUBIK building to capture real-time data and store it in a persistent environment. The stored data is analyzed to find potential solutions to automate different processes. Each connected device uses a standardized communication protocol to enable interoperability among them. A hub acts as a middleware between connected devices and external software programs and manages the communication protocol. The communication protocol may vary because of the installed devices' connectivity and cause additional complexity for large deployment scenarios. Therefore, it is necessary to create a general-purpose system to centralize the connections no matter what device type is connected.

Table 11.2. Devices/Signal on Ground Floor and Reflected Ceiling of KUBIK Building. (K) values represents Kitchen, (B) values represents Bedroom, (L) value represents Living Room, (C) value represents corridor.

ID	Type	Device type	Location	System	Signal
9	Sensor	Ceiling Multisensor 1	K	Z-Wave	Motion: YES/NO
9	Sensor	Ceiling Multisensor 1	K	Z-Wave	Temperature: degrees Celsius
9	Sensor	Ceiling Multisensor 1	K	Z-Wave	Light: 0 lux–1000 lux
9	Sensor	Ceiling Multisensor 1	K	Z-Wave	Relative humidity: 20%–95%
10	Sensor	Ceiling Multisensor 2	L	Z-Wave	Motion: YES/NO
10	Sensor	Ceiling Multisensor 2	L	Z-Wave	Temperature: degrees celsius
10	Sensor	Ceiling Multisensor 2	L	Z-Wave	Light: 0 lux – 1000 lux
10	Sensor	Ceiling Multisensor 2	L	Z-Wave	Relative humidity: 20%–95%
11	Sensor	Ceiling Multisensor 3	L/C	Z-Wave	Motion: YES/NO
11	Sensor	Ceiling Multisensor 3	L/C	Z-Wave	Temperature: degrees Celsius
11	Sensor	Ceiling Multisensor 3	L/C	Z-Wave	Light: 0 lux–1000 lux
11	Sensor	Ceiling Multisensor 3	L/C	Z-Wave	Relative humidity: 20%–95%
12	Sensor	Ceiling Multisensor 4	B	Z-Wave	Motion: YES/NO
12	Sensor	Ceiling Multisensor 4	B	Z-Wave	Temperature: degrees Celsius
12	Sensor	Ceiling Multisensor 4	B	Z-Wave	Light: 0 lux – 1000 lux
12	Sensor	Ceiling Multisensor 4	B	Z-Wave	Relative humidity: 20%–95%
13	Sensor	Smoke Detector 1	K/L/C	Z-Wave	Alarm state: ON/OFF
14	Sensor	Smoke Detector 2	B	Z-Wave	Alarm state: ON/OFF
15	Actuator	Ceiling light 1	K	PLC	Light state: ON/OFF
16	Actuator	Ceiling light 2	L	PLC	Light state: ON/OFF
17	Actuator	Ceiling light 3	B	PLC	Light state: ON/OFF
18	Actuator	Fan Coil 1	K	PLC	Operation state: ON/OFF
18	Actuator	Fan Coil 1	K	PLC	Temperature setpoint: Celsius
19	Actuator	Fan Coil 2	L	PLC	Operation state: ON/OFF
19	Actuator	Fan Coil 2	L	PLC	Temperature setpoint: Celsius
20	Actuator	Fan Coil 3	L	PLC	Operation state: ON/OFF
20	Actuator	Fan Coil 3	L	PLC	Temperature setpoint: Celsius
21	Actuator	Fan Coil 4	L	PLC	Operation state: ON/OFF
21	Actuator	Fan Coil 4	L	PLC	Temperature setpoint: Celsius
22	Actuator	Fan Coil 5	C	PLC	Operation state: ON/OFF
22	Actuator	Fan Coil 5	C	PLC	Temperature setpoint: Celsius
23	Actuator	Fan Coil 6	B	PLC	Operation state: ON/OFF
23	Actuator	Fan Coil 6	B	PLC	Temperature setpoint: Celsius
24	Actuator	Fan Coil 7	B	PLC	Operation state: ON/OFF
24	Actuator	Fan Coil 7	B	PLC	Temperature setpoint: Celsius

The devices installed in KUBIK use two different communication protocols. The first one is called Z-Wave,[1] a wireless communication protocol that integrates smart sensors inside a building. Z-Wave devices are widely used in domestic environments due to their ease of installation and low cost. However, its signal quality can be affected by interference and its battery level. The second one is called the MODBUS[2] communication protocol. This protocol is an industry-standard that is robust, fast, and secure. The connected devices using the MODBUS communication protocol need a central node called *industrial PC* or Programmable Logic controller.[3] They need to be connected by a physical connection (a wired cable). The MODBUS communication protocol is widely used in industrial processes to obtain information from machines and active processes. Thus, the main difference between a Z-Wave and a PLC device is that the former does not need any physical connection, and the latter requires a physical connection to an industrial PC. In terms of installation, Z-Wave-based devices are more convenient than a PLC device, but a PLC device offers robust connectivity and high reliable data speeds.

Having two different communication protocols to acquire data or perform actuation processes, we need to implement a middleware that enables interoperability among various sensors and actuators. The SMOOL IoT middleware that has a semantic broker for connecting heterogeneous devices or sources of information. In addition, the Building Management System also centralize all the information of new wireless sensors/actuators and the legacy building control systems of the building using a Scada software.

To address the challenges of sharing sensors/actuators between IoT applications that are running at the same time and also add a security layer to the data streams, the ENACT project provides tools and enablers to ensure the trustworthiness of the IoT applications in the KUBIK infrastructure. Figure 11.4 depicts the high-level architecture of the communications architecture in the KUBIK building.

The high-level architecture of the communications in KUBIK building is divided into three modules. The first module contains two subsystems: (i) **system 1**, the *wireless* system where each device is connected to a central node or *network hub* managing wireless connections; and (ii) **system 2**, a *wired* system where each device is physically connected to a central node or PLC device managing each wired connection. The second module contains the Building Management System (BMS) having three main elements: (i) a gateway device managing the connections between PLC/Z-Wave nodes, (ii) a persistent database that gathers the

1. https://www.z-wave.com/
2. https://modbus.org/
3. https://en.wikipedia.org/wiki/Programmable_logic_controller

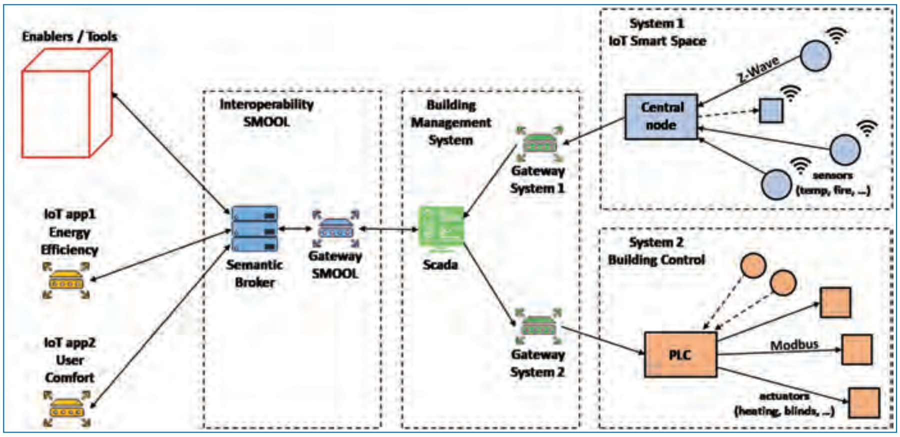


Figure 11.4. High level communications architecture in the KUBIK building.

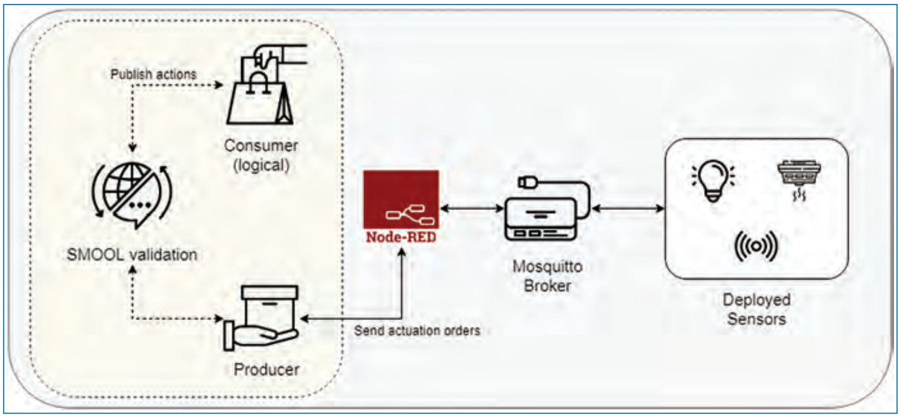
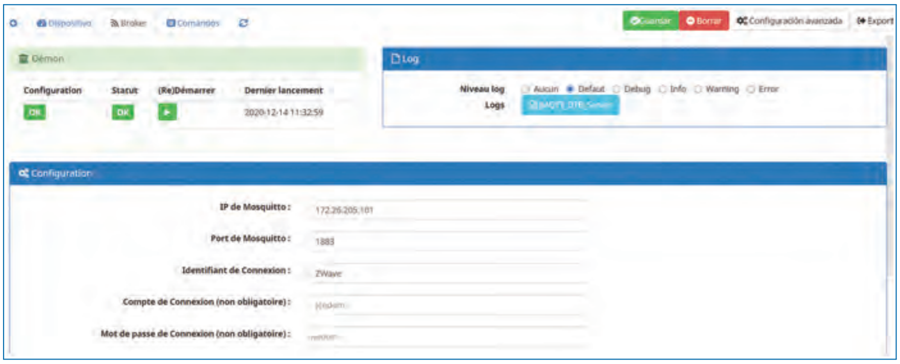


Figure 11.5. Technical components used in KUBIK architecture.

total connected devices and stores the data read by each device, (iii) a SCADA interface that graphically shows the current status of each deployed device and enables engineers to update device configurations. The last module is the SMOOL interoperabilty layer [2]. The entire system is managed by the decisions made according to a set of rules programmed in the IoT applications.

Figure 11.5 depicts the technical components used in the above architecture. 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) obtain data measures to be sent to a middle device called "Mosquitto MQTT broker".[4]

4. https://mosquitto.org/



Configuration: OK
Statut: OK
(Re)Démarrer
Dernier lancement: 2020-12-14 11:32:59

Niveau log: Aucun [✓ Defaut] Debug Info Warning Error
Logs: jMQTT_DTB_Sensor

IP de Mosquitto : 172.26.205.101
Port de Mosquitto : 1883
Identifiant de Connexion : ZWave
Compte de Connexion (non obligatoire) : Jeedom
Mot de passe de Connexion (non obligatoire) : [illegible]

Figure 11.6. Jeedom configuration interface.

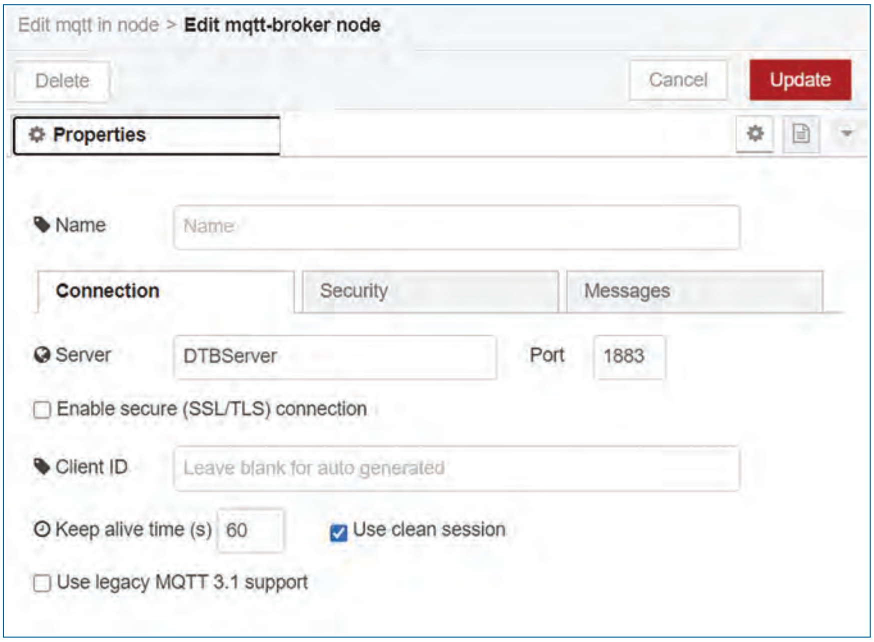
The SMOOL middleware has two components, i.e., the Producer and Consumer modules. The Producer module sends secure actuations when the Consumer module requires a security token. The Consumer module executes a set of expert rules using the obtained data from the Producer module. An expert encodes the expert rules. He/she decides which recommended actions to take when the data acquired from sensors meets a condition. For example, if the illumination sensors detect too much light, the actuators open the blinds for sunlight to get inside the KUBIK's living room.

At the low-level in the smart building Architecture (Figure 11.5), the data acquisition and actuation processes are managed by some hub systems called Gateways. These hubs are configured to allow direct communication between different sensors/actuators and a Mosquitto MQTT broker. Each hub uses its communication protocol to acquire or send actuation orders to the target device. For example, one hub is configured to manage only Z-Wave communications towards connected devices, while another hub is configured with the MODBUS communication standard. These hub systems use an internal operating system, JEEDOM,[5] that enables a graphical configuration of the connected devices and external services. In this regard, each hub is configured to make a direct connection to the Mosquitto MQTT broker. Figure 11.6 shows how JEEDOM is configured to send the data read from a set of Z-Wave devices directly to the Mosquitto MQTT broker.

Once the connection between the Z-Wave/PLC hub (JEEDOM) and Mosquitto broker is established, the next step is to program the Node-RED programming tool to develop a bridge between MQTT messages, SMOOL middleware and logic of IoT applications. Node-RED[6] is a visual flow-based programming environment

5. https://www.jeedom.com/site/en/index.html

6. https://nodered.org/


Edit mqtt in node > **Edit mqtt-broker node**

Delete Cancel Update

Properties

Name: Name

Connection | Security | Messages

Server: DTBServer Port: 1883

☐ Enable secure (SSL/TLS) connection

Client ID: Leave blank for auto generated

Keep alive time (s): 60 ☑ Use clean session

☐ Use legacy MQTT 3.1 support

Figure 11.7. Node-RED programming to connect to Mosquitto MQTT Broker.

designed by IBM for the Internet of Things. Figure 11.7 depicts the configuration parameters needed to connect the Node-RED with the Mosquitto broker. These parameters enable the configuration of different workflows to make the required subscriptions to each connected device.

After having the logical connection between Node-RED and Mosquitto, it is necessary to program the required Node-RED flows for the data acquisition process with the SMOOL module. Figure 11.8 shows the acquisition process flow programmed in Node-RED. It reads the data from the Mosquitto broker and exposes it directly to its internal REST API module. The Producer SMOOL component reads the data exposed in the Rest API endpoints. Each connected line in Figure 11.8 represents a device inside the KUBIK building. There are several sensors deployed in the KUBIK building, and each one has its action flow. For convenience, Figure 11.8 presents only a minor part of the devices.

Figure 11.9 exhibits the secure actuation process provided by SMOOL middleware having two action flows: (i) one flow to send the available orders from Node-RED to SMOOL using a security token (top of the image) and (ii) another flow to sent the secure actuation orders checked in SMOOL to the actuator via Mosquito broker (low part of the image).

The flows are configured to enable the SMOOL components to send actions to the hub and write the needed information to perform the move up or move down actuation orders of the living room/kitchen blinds.

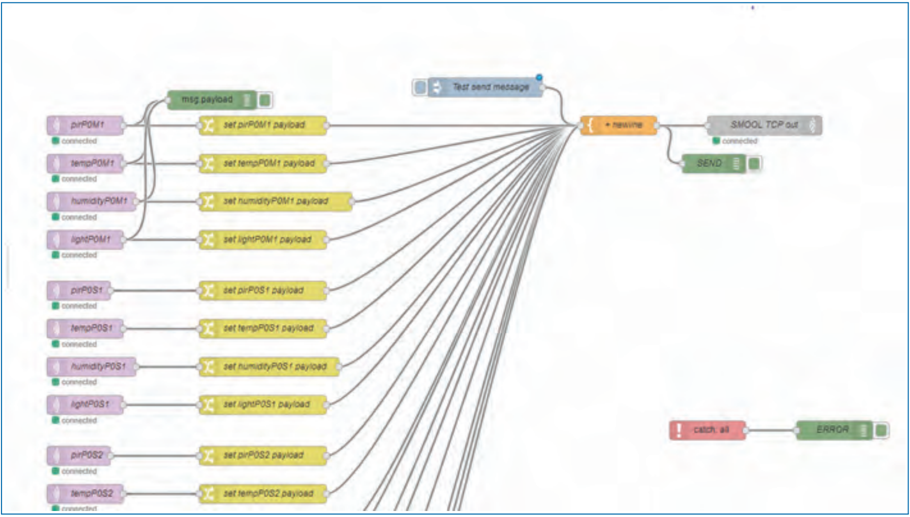


Figure 11.8. Node-RED flows to publish sensorized data in SMOOL.

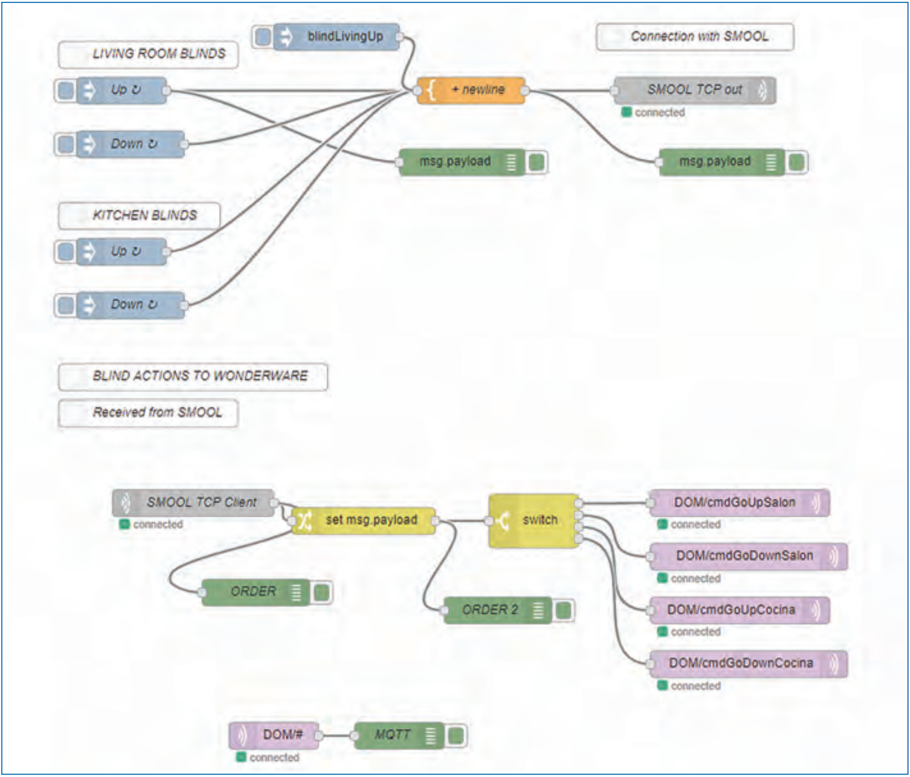


Figure 11.9. Node-RED flows to execute orders over devices in KUBIK through SMOOL.

11.4 KUBIK as an Experimental Platform for the ENACT Scenarios

In the ENACT project, the KUBIK building has been used as an experimental platform to test enablers in the *Smart Building* use case. Multiple scenarios have been implemented to test the ENACT tools relevant to this use case. We give the details of the scenarios and their main results in the following section. Those scenarios involve concurrent IoT applications related to energy efficiency and user comfort. These applications implement thermal comfort controls and smart building alerts that assure the safety of their occupants. They employ Z-wave and PLC systems described in the previous section.

11.4.1 Scenario 1: Thermal Comfort Control – Heating Design

Thermal comfort control is an essential smart building function. Sensors measure the users' comfort and enable the HVAC control system to keep the temperature at the level requested. By adding thermostat and temperature sensors with another protocol than the one of the HVAC system, it is easier to retrofit old systems and give more flexibility to deploy sensors. On the other hand, this strategy poses potential threats and risks to the thermal comfort system that must be identified and addressed.

One of the threats identified is when one of the thermal control devices is replaced with a similar device but not the same one. The Risk Management enabler is then used to analyse potential threats to the HVAC control system when new IoT devices are combined with legacy systems, provide the list of mitigation actions that the new IoT device needs to fulfil, and support the selection of security controls to minimize risks. By means of this enabler, HVAC control system designers and maintainers can also decide the risk level that that it is tolerable.

The retrofitting of old HVAC control systems is usually combined with changing the logic of the HVAC control program. In addition, the SMOOL middleware is used to communicate with new IoT devices and ThingML language can program those devices to define system behaviours and generation of executable code. The Orchestration and Continuous Deployment enabler, aka. GeneSIS, enables the continuous deployment and update of applications. In the ENACT project, GeneSIS has been fully integrated with SMOOL semantic middleware and ThingML language. In that way, SMOOL and ThingML are automatically deployed by GeneSIS as any other software component when adding new IoT devices and changing the HVAC control logic. The programming of the HVAC control logic, the communication characteristics of the new IoT devices in SMOOL and the programming of ThingML devices are done as part of the same project in the same Integrated Development Environment (IDE).[7]

The Orchestration and Continuous Deployment enabler has also been used with the Actuation Conflict Manager enabler in Scenario 2.

11.4.2 Scenario 2: Thermal Comfort Control – Conflict in Heating Actuator Use

In the continuation of scenario 1, the integration of multiple systems can result in two or more applications sending different temperature preferences to the same heating actuator, which causes a fluctuating operation of the thermal control. The Actuation Conflict Management (ACM) enabler allows findings at design time the actuation conflicts that may lead to these fluctuations, and then it helps solving the conflict, thereby fixing the cause of the fluctuation.

A direct conflict occurs when two applications try to access the same node, e.g. an actuator. These two applications accessing the same actuator might send contradictory commands resulting in an indeterministic behaviour. An example of direct actuation conflict has been programmed and tested in the ACM enabler (see Figure 11.10). Figure 11.10 shows two IoT applications fed with temperature values from two different sensors. The applications try to change the actuator state when the temperature values reach a threshold that mimics the thermostat operation. The temperature sensors are deployed at different parts of the room, and thus different temperature values are likely obtained. The actuator behavior is similar to a relay that can switch ON and OFF the thermal heater.

Figure 11.11 shows a Node-RED flow with the implementation of the previously described scenario. In that flow, two identical subflows represent sensor processes that access the "command" node to send an actuation command to the same heating

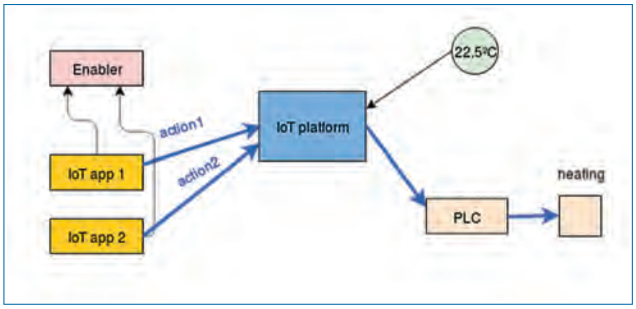


Figure 11.10. Thermal comfort control with conflict in heating actuator use.

7. Details about this integration can be seen in the following video https://www.youtube.com/watch?v=mfT_AwfkXNc

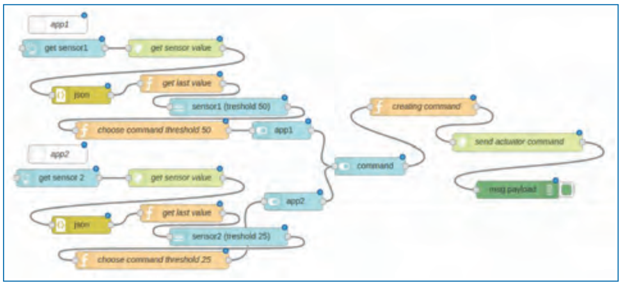


Figure 11.11. Node-RED flow example for thermal comfort control to be used by the ACM enabler.

actuator. The ACM enabler helps the software developer solve actuation conflicts occurring when two or more concurrent IoT applications try to send simultaneously conflicting actuation orders to the same actuator. In this scenario, a docker component that contains the Node-RED flows of the IoT application is deployed using GeneSIS. The ACM enabler then imports the model of the IoT system created by GeneSIS. GeneSIS enables the creation of an architecture and deployment model of the Smart IoT System by adding components and links. Although several model formats can be imported into the ACM enabler, Node-RED and GeneSIS are the main tools supported by the ACM enabler (Figure 11.11).

Once everything is imported and set up in the ACM enabler, a click on the "find conflicts" button automatically detects actuation conflicts and add a placeholder for actuation conflict management component in the model where conflicts might happen. The ACM enabler proposes several out-of-the-box components to solve a detected actuation conflict. The user chooses the right component, and the actuation conflict management placeholder is automatically replaced accordingly. Then, the IoT application is updated to become a conflict-free thermal comfort control system that can be redeployed using GeneSIS.

11.4.3 Scenario 3: Luminosity Comfort Control – Indirect Conflict in Luminosity Level Actuation

Two or more applications may also send actuation orders to different actuators that cause an indirect actuation conflict. An indirect actuation conflict affects the building thermal control of a building when two IoT applications managing two actuators act concurrently over the same physical variable, e.g. setting a high-temperature setpoint to the HVAC and opening a window. For instance, one app opens the window (lower temperature), and another one increases the setpoint temperature in the HVAC control system (higher temperature). An indirect actuation conflict can also occur in the room's luminosity level control when acting on lights and blinds. One switch controls the light to set it set ON or OFF, and another one controls the blind

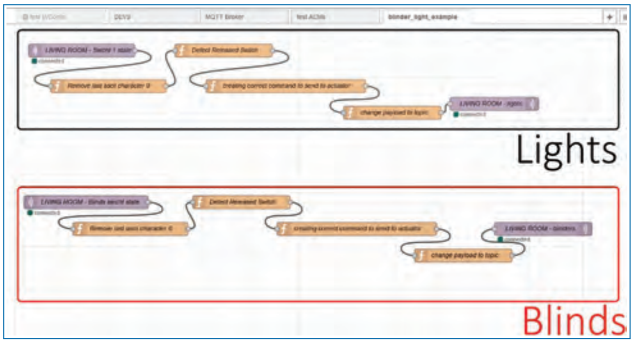


Figure 11.12. Node-RED flow representing the classic behaviour of the control of lights and blinds using switches.

to set it UP or DOWN. A physical model of the building is needed to find the indirect conflict on brightness, i.e. the impact of lights and blinds on the luminosity level.

The Actuation Conflict Management (ACM) enabler also corrects the operation of a physical system subject to uncertainties to deal with indirect conflicts. To achieve that, the physical system model is added to the enabler to detect and solve such indirect actuation conflicts. The ACM enabler imports the IoT system model created using GeneSIS and Node-RED tools. Once the model is imported, a physical process configuration is specified. The configuration allows establishing the interaction between a logical node and the environment. For instance, when there is an activation of a light, it is linked to a physical process representing luminosity. The utility of this process is to find an indirect conflict. If two different actions are linked to the same physical process, it may be an unplanned conflict to be solved.

In this scenario, the control of lights and blinds are associated with their corresponding switches using Node-RED. The classic behaviour of the system is that the lights are controlled by one switch to set it ON or OFF and the blinds are controlled by another switch to set it UP or DOWN (see Figure 11.12).

The ACM enabler then is fed with the previous described Node-RED flow for classic behaviour and the physical representation of the system to find actuation conflicts, i.e. the impact of lights and blinds on the luminosity level. Then, the ACM tool detects an actuation conflict in the luminosity level when we turn on the lights having sufficient brightness outside. The ACM tool creates a new Node-RED flow that resolves that conflict by adding new ACM components between the application logic and the actuation command (see Figure 11.13). The modified new Node-RED flow for luminosity level control can then be redeployed. The updated behaviour open the blinds instead of turning on the lights when the outside luminosity is high enough. Also in this sceanario, the Behavioural Drift Analysis

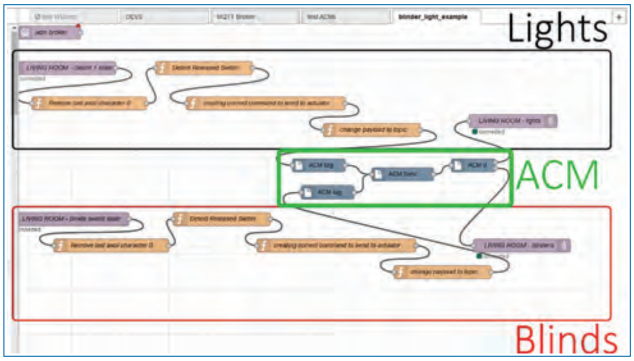


Figure 11.13. Node-RED flow example for solving the indirect conflict in luminosity level actuation.

(BDA) enabler was used to evaluate how much the observed behaviours of the IoT System are different from the expected ones.

11.4.4 Scenario 4: Smart Building Alerts for User Comfort

Data integrity and confidentiality must be ensured during data communication, especially if the communication is an important alert about equipment and user safety. Therefore, tampering with the alarms by external people should be avoided. The Security & Privacy Monitoring & Control (S&P Mon&Con enabler, cf. Chapter 7.1) enabler ensures the integrity and privacy of the communications through two services: (i) the Security and Privacy Monitoring for the surveillance of data security and (ii) the Security and Privacy Adaptation for data security enforcement.

This scenario addresses smart building alerts for IoT apps monitoring building aspects, such as an abnormally high or low temperature and smoke presence. The Security and Privacy Monitoring and Control (S&P Mon&Con) enabler ensures that the alarm system is not tampered with by external people and safeguard people's privacy. These enablers have been applied to the communication architecture in Figure 11.4.

The Security & Privacy Monitoring & Control enabler uses port mirroring to send a copy of the network packets that contain sensitive information of the smart building to a server where the enabler is running. Three different nodes in the smart building communication architecture have been mirrored: (i) the SMOOL communication broker, (ii) the Raspberry Pi used as a gateway by the SMOOL, and (iii) the SCADA node of the Building Management System of the KUBIK building.

When sensing information is transmitted, the S&P Mon&Con enabler checks if the messages are sent by the authorized nodes that belong to the KUBIK network. If an external device tries to send or receive a message to/from the monitored nodes, the S&P Mon&Con enabler notifies the system administrator to block network traffic if necessary. The S&P Mon&Con enabler also ensures the sensing data integrity by blocking any attempt of tampering with the message.

Scenario 4 addresses sensor devices that trigger alarms. These alarms may also trigger a security action or siren. The security actions consist of an ON/OFF command to a siren.

11.4.5 Scenario 5: Thermal Comfort Control – Self-optimizing Controller Design

The thermal comfort control of a building is a trade-off between user comfort and energy consumption, e.g., the heating is off when the room or building is not occupied. Monitoring the building's thermal inertia and real-time occupation can significantly improve energy efficiency and comfort compared to traditional controllers. We used the Online Learning enabler (OLE) to find and update the optimal control parameters at run time. We employed GeneSIS to properly deploy those control parameters and orchestrate the involved IoT devices.

The Online Learning (OLE) enabler is a module with an agent-based Artificial Intelligence (AI) algorithm that performs actions based on the sensors readings gathered from the KUBIK building infrastructure. This enabler aims to provide a thermal comfort solution that reduces the energy costs in the building. We have developed a thermal model of the ground floor of the KUBIK building to implement the OLE. We have demonstrated the potential for energy saving and increased thermal comfort.

11.5 Conclusion

Buildings are becoming increasingly smart, and new IoT tools are needed to ensure the safe and trustworthy operation of the different services they offer. The KUBIK experimental building is a smart building testbed for the ENACT project tools.

We successfully tested the ENACT enablers that enable continuous deployment, solving actuation conflicts, ensure security and privacy of the communications, identify risks at design time, correct behavioral drifts, enforce security and privacy, and self-optimizing controller.

References

[1] José A. Chica *et al.* “Kubik: Open Building Approach for the Construction of an Unique Experimental Facility Aimed to Improve Energy Efficiency in Buildings”. In: *Open House International* 36.1 (Jan. 2011), pp. 63–72. ISSN: 0168-2601. DOI: 10.1108/OHI-01-2011-B0008.

[2] Adrian Noguero, Angel Rego, and Stefan Schuster. “Towards a smart applications development framework”. In: *Social Media and Publicity* 27 (2014), pp. 2011–2020.

DOI: 10.1561/9781680838251.ch12

Chapter 12

Looking Ahead

By Andreas Metzger, Cristóbal Costa Soria, Juan Garbajosa, Ana M. Moreno, Daniel Pakkala, Jukka Rantala, Valère Robin, Jukka Saarinen, Bjørn Skjellaug, Hui Song, Mike Surridge, Tuomo Tuikka, Josef Urban and Thorsten Weyer

12.1 Introduction

In this book, we reported the main outcomes of the EU Horizon 2020 project ENACT. ENACT developed a toolkit that facilitates the development and operations (DevOps) of trustworthy Smart IoT Systems.

Concluding this book, we look ahead and offer perspectives on future research and innovation opportunities in the area of Smart IoT Systems. These future research and innovation opportunities are based on research challenges that were jointly developed with partners from the European Technology Platform *NESSI* (Networked European Software and Services Initiative [1]). These research challenges were contributed as input to the forthcoming European Key Digital Technologies (KDT) Partnership. We thus use the term *KDT applications* as an umbrella term for Smart IoT Systems, and thereby include closely related areas such as embedded systems, cyber-physical systems and edge-based systems.

12.2 Research and Innovation Opportunities

12.2.1 Software-driven Integration of KDT Applications

Without interaction with each other, KDT applications are information silos, which become an obstacle for potential business value creation. Software-driven integration, *i.e.*, developing new software to integrate existing applications, is a trend in the software industry. Mainstream cloud applications already underwent platformization, *e.g.*, Facebook or Salesforce are now platforms that allow third parties to develop and offer value-added services.

Because platformization of KDT applications requires offering powerful APIs to external components and systems, new research and innovation into such platforms is needed. Platformization also requires novel kinds of software architectures within the applications to achieve the flexibility for deep customization. Supporting integration is challenging because custom code will share the already constrained resource of electronic components and may also bring vulnerability to applications that are security- or safety-critical. New design methods are needed, with performance and vulnerability assessment considering resource and hardware aspects, together with novel isolation mechanisms on low-level electronic components, potentially supported by virtualization techniques.

KDT applications also need to be integrated with traditional enterprise and consumer software applications, since the latter are currently managing the data and business processes. Such integration introduces electronic components into the traditional human-data interaction, and thus calls for novel software-hardware codesign in an agile and continuous way, in order to bridge the social-cyber-physical sensing powered by new electronic components with the business data and process controlled by the traditional enterprise software systems. This integration with traditional enterprise and consumer software applications, also require a higher-level analysis of vulnerability and risks. Organizations try to run high level business processes over inefficient digital ecosystems, made out of different parts, some of them new, some of them legacy-based, and in many cases non-interoperable. There is a lack of mechanisms to collect information about these processes (transversal analytics) and control risks from data pieces coming from different hardware and software components.

Software integrators play an important role in the software industry and in the software value chain. New abstractions, orchestration languages and integration methodologies are needed to facilitate the interdisciplinary thinking of integrators. Future research and innovation actions in this direction require close collaboration among software engineering researchers, application providers, software integrators and the integration platform providers, resulting in the extension of mainstream iPaaS (integration Platform as a Service) solutions to reach the lower-end devices. Platformization and integration are also important for KDT application providers to develop ecosystems involving component providers, software vendors and integrators. Research and innovation is needed to investigate business models, legal issues, data sharing strategies, etc., in order to integrate scattered businesses into prosperous European KDT ecosystems.

12.2.1.1 Managing complexity, dynamics, and uncertainty of KDT applications

Digital systems and groups of collaborating systems, as well as the environments in which these systems operate, are becoming more complex, show highly dynamic behaviour, and increasingly face uncertainty during operation. The trend toward digitalization is accompanied by a significant increase in complexity, dynamics and uncertainty. The increasing complexity can be seen in individual KDT systems, groups of collaborating systems, as well as the environments in which a particular KDT system operates.

Managing the dynamics of systems, collaborating groups and their environment poses an important challenges for the engineering KDT applications. Furthermore, KDT applications increasingly face uncertainty during runtime, *i.e.*, KDT applications have incomplete or ambiguous information about the environment in which they operate. This is especially the case when systems are operating in open contexts where the relevant properties of the environment cannot be completely anticipated at design-time, and therefore cannot be fully handled by predefined adaptations. In many future scenarios, such as autonomous driving or smart factories, systems must be able to meet their goals even on the basis of incomplete or contradictory information about the environment, *e.g.*, the intentions of other systems or humans, or the preferences and skills of human users.

While ENACT has indicated how – via machine learning at runtime (see Chapter 6) – a KDT application may capture uncertainties in the environment, additional research is needed to answer questions such as how to guarantee appropriate and safe cognitive adaptability in complex, highly dynamic and uncertain environments, and how to verify – at runtime and under hard real-time constraints – emergent system behavior resulting from the interactions of subsystems and the ambiguity faced in such environments.

A central challenge will be the modelling and implementation of human-machine interactions under those conditions; for example the efficient and effective transfer of control between systems and human users to avoid effects of mode confusion in autonomous driving. Developing software for complex and dynamic systems, able to deal with uncertainties, will require engineering processes involving multiple disciplines such as cognitive science and sociology.

Promising approaches to deal with the increasing complexity, dynamics and uncertainty must consider both design-time and runtime. Such approaches will be based on innovative combinations and improvements of software technologies in the following categories:

- innovative technical solutions in the area of software technology including design-time and runtime techniques for collaborative information fusion;
- collaborative runtime verification (including digital twin technology);
- environment perception with shared models of the environment;
- prediction of future behaviour and cognitive adaptability of individual systems, collaborative groups and the environment.

Innovative process-related solutions in software technology should include approaches from design science, agile methods, and the creation of appropriate team culture. Design science offers a disciplined approach to analyze a problem in a real-world context, systematically derive solutions (in the form of software artefacts and prototypes) and validate and evaluate them in order to generate new knowledge [2]. In design science digital solutions are produced and studied in an operational application context (real world use context), where the maturity, quality and value of the solutions can be evaluated. The solutions often involve combinations of evolving operational processes, software, hardware and ICT systems governed by multiple organizations, which creates a complex context for design and deployment of new digital solutions. At design- time, the related engineering activities, including software engineering, need to be well aligned with the overall solution goals and requirements, and hardware and network requirements and limitations, as well as business requirements, need to be considered in parallel with the software engineering process. At runtime, the resulting technical system (or system-of-systems) may be widely distributed across different embedded, networked and cloud computing nodes governed by multiple different organizations. Accordingly, to discover ways to manage the complexities involved in design and deployment of new digital solutions, the role of governance boundaries and multi-organization collaboration, both at design-time and runtime, deserve further research.

12.2.2 Leveraging Spatial Computing for KDT Applications

Spatial Computing is an emerging interaction mechanism for digital content in a converged cyber-physical world. Advances in devices and user interfaces (*e.g.*, mixed reality glasses, gesture recognition, haptic feedback, interfaces built from new materials) and their integration into a spatial computing system will allow for more adaptable, responsive and immersive interactions with the digital world. The services offered by the spatial computing system will provide new ways of augmenting user experience and will allow us to 'feel' with all senses the virtual environment around us. Just like the real word, spatial computing offers a rich environment for multi-user interaction, and empowers a human-centric approach for future digitalisation. Taking industrial automation as an example, the introduction of spatial computing will help put human needs and interests in a central role, focusing not only on how to automate and optimize the production process, but also on how to get workers involved in the process.

Spatial computing systems will be complex constellations of software and content components operated by a multitude of ecosystem participants. The intelligence required for smart interaction mechanisms will depend on collecting and analysing massive amounts of social cyber-physical sensing data across all stakeholders in the ecosystems into digital super twins. The required computing power will not be provided by the involved devices only, and thus limited to their capabilities – it will be provided by the cloud or at the edge also, offering additional and typically more powerful capabilities.

Software engineering approaches for designing and developing spatial computing systems will need to cope with the challenging environment of diversity in devices and application domains, with intelligent deployment and adaptation capabilities. New programming models, languages and methodologies will emerge in the spatial computing era, and research and innovation actions will help to create breakthrough progresses. This will require interdisciplinary research integrating advances in media technologies (new coding technologies for digital content), cognitive psychology (new human-machine interaction for better attention and perception from the users), social science (new methods and processes for better human collaboration), etc.

12.2.3 Sustainable and Energy-efficient KDT Applications

Although digital technologies and software may provide very powerful tools to optimize the energy efficiency in vertical domains, their absolute and relative energy and resource consumptions continue to increase, even if hardware itself improves (notably for embedded systems as a by-product of autonomy optimisation or to reduce the cost of big data centres). The increasing functional scope of software and applications, the introduction of data intensive algorithms and systematic logging of events, the use of complex middleware stacks (hypervisors, virtual machines, containers, languages runtimes, bloated framework) all contribute to the environmental impact of software-based systems, often sacrificing frugality for the sake of ease of development and time-to-market.

The sustainability concern needs to be natively addressed in the development and execution phase of all digital systems (embedded, personal, large-scale, communication equipment, etc.). New tools and models are needed to optimize the interactions between hardware and lower software layers, to adapt to runtime context, and continuously minimize energy and resource consumption, based on monitoring not only the internal behaviour of software systems but also the external physical environment through advanced sensing and learning.

Sustainability also calls for simplified and efficient architectural patterns, along with the appropriate education of key actors such as developers, software architects, system integrators and data centre management teams. Interdisciplinary research will provide novel solutions towards sustainable digital systems, *e.g.*, the use of energy harvesting from the environment (wind, solar, pressure, etc.) or other energy sources (body heat, foot strikes, etc.) to power lower-end devices, making them battery-free. This also calls for new programming models, software architectures and self-adaptation approaches to cope with novel and potentially unstable power sources.

12.3 Conclusion

The ENACT project paved the way towards bringing established software engineering processes and tools to the realm of Smart IoT Systems and KDT applications. As indicated above, this is a mere start and challenging research and innovation opportunities are ahead of us. Jointly addressing these challenges will contribute to European companies, SMEs, and research institutes to remain competitive in this traditionally strong area of Europe.

References

[1] NESSI ETP. *Software and Key Digital Technologies*. Networked European Software and Services Initiative, Brussels. (http://www.nessi.eu/)

[2] R. Wieringa. *Design Science Methodology for Information Systems and Software Engineering* Springer, 2014.

Index

About the Editors

Nicolas Ferry is an Associate Professor at University Côte d'Azur. Prior he was a Senior Research Scientist at SINTEF. He holds a Ph.D. degree from the University of Nice. His research interest includes model-driven engineering, domain-specific languages, Internet of Things, cloud-computing, self-adaptive systems, and dynamic adaptive systems. He has actively contributed to various national and international research projects such as the REMICS, CITI-SENSE, MC-Suite and MODAClouds EU projects, and is the technical manager of the H2020 ENACT project. He has also served as a program committee member of international conferences and workshops.

Erkuden Rios received the B.S. and M.S. degree in telecommunication engineering from University of Basque Country, Bilbao, Spain, in 1997. In 2020 Erkuden has received a Ph.D. in multi-Cloud security assurance from University of Basque Country. After working six years for Ericsson Spain, she is currently senior scientist of Cybersecurity research team of ICT Division in Fundación Tecnalia Research & Innovation, Derio, Spain. She is currently the coordinator of the Security WP in the H2020 ENACT project on Secure and Privacy-aware Smart IoT Systems as well as in the H2020 SPEAR project on Secure Smart Grids. Previously, she was the coordinator of the H2020 MUSA project on Multi-cloud Security as well as the chair of the Data Protection, Security and Privacy in Cloud Cluster of EU-funded research projects, launched by DG-CNECT in April 2015. (https://eucloudclusters.wordpress.com/data-protection-security-and-privacy-in-the-cloud/).
Furthermore, she has worked in multiple large European and Spanish projects on cybersecurity and trust such as POSEIDON, PDP4E, TACIT, RISC, ANIKETOS, SWEPT, CIPHER and SHIELDS. Her main research interests include Trust and Security, Risk Management, and AI for Cybersecurity. Mrs. Erkuden collaborates with Technology Platforms and Forums such as Cybersecurity PPP ECSO, ETSI Secure Artificial Intelligence Working Group, AIOTI WG4 Policy and Privacy and

the Spanish National Network on Cybersecurity. She has been member of Programme Committees of Journals and Conferences.

Andreas Metzger is senior academic councilor at the University of Duisburg-Essen and heads the Adaptive Systems and Big Data Applications group at paluno, the Ruhr Institute for Software Technology. He holds a Ph.D. in computer science from the Technical University of Kaiserslautern. His background and research interests are software engineering and machine learning for self-adaptive systems. Among other leadership roles, he was technical coordinator of the European H2020 lighthouse project TransformingTransport and work package leader in the H2020 ENACT project. Andreas serves as steering committee vice chair of NESSI, the European Technology Platform dedicated to Software, Services and Data, and as deputy secretary general of the Big Data Value Association (BDVA/DAIRO).

Hui Song is a senior researcher with SINTEF Digital, Norway. He has a Ph.D. in computer science from Peking University in China. His research interests include software engineering methods and tools, and their applications on developing cloud and IoT systems. He has contributed to a number of European and Norwegian research projects, and is the coordinator of the H2020 ENACT project.

Contributing Authors

Miguel Ángel Antón
TECNALIA, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastián, Spain
mangel.anton@tecnalia.com

Larraitz Aranburu
TECNALIA, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastián, Spain

Franck Chauvel
SINTEF Digital, Oslo, Norway
franck.chauvel@sintef.no

Tommaso Crepax
KU Leuven, Belgium
tommaso.crepax@kuleuven.be

Rustem Dautov
SINTEF Digital, Oslo, Norway
rustem.dautov@sintef.no

Franck Dechavanne
Université Côte d'Azur, I3S/CNRS Sparks, Sophia Antipolis, France
franck.dechavanne@sintef.no

Jacek Dominiak
Beawre, Barcelona, Spain
jacek.dominiak@beawre.com

Felix Feit
paluno – The Ruhr Institute for Software Technology University of Duisburg-Essen, Essen, Germany
felix.feit@paluno.uni-due.de

Nicolas Ferry
Université Cote d'Azur, I3S/INRIA Kairos, Sophia Antipolis, France
nicolas.ferry@univ-cotedazur.fr

Franck Fleurey
Tellu, Asker, Norway
franck.fleurey@tellu.no

Anne Gallon
Evidian – ATOS, France
anne.gallon@evidian.com

Juan Garbajosa
UPM, Spain

Thibaut Gonnin
Université Côte d'Azur, I3S/CNRS Sparks, Sophia Antipolis, France
thibaut.gonnin@univ-cotedazur.fr

Elena Gonzalez-Vidal
Beawre, Barcelona, Spain
elena.gonzalez-vidal@beawre.com

Modris Greitans
Institute of Electronics and Computer Science (EDI), Riga, Latvia
modris_greitans@edi.lv

Christophe Guionneau
Evidian – ATOS, France
christophe.guionneau@evidian.com

Vinh Hoa La
Montimage, Paris, France
vinh.hoala@montimage.com

Eider Iturbe
TECNALIA, Basque Research and Technology Alliance (BRTA), Derio, Spain
eider.iturbe@tecnalia.com

Sergio Jimenez Gomez
Indra Sistemas S.A, Madrid, Spain
sjimenez@indra.es

Janis Judvaitis
Institute of Electronics and Computer Science (EDI), Riga, Latvia
janis.judvaitis@edi.lv

Stéphane Lavirotte
Université Côte d'Azur, I3S/CNRS Sparks, Sophia Antipolis, France
stephane.lavirotte@univ-cotedazur.fr

Wissam Mallouli
Montimage, Paris, France
wissam.mallouli@montimage.com

Saturnino Martinez
TECNALIA, Basque Research and Technology Alliance (BRTA), Derio, Spain
saturnino.martinez@tecnalia.com

Andreas Metzger
paluno – The Ruhr Institute for Software Technology University of Duisburg-Essen, Essen, Germany
andreas.metzgger@paluno.uni-due.de

Yuliya Miadzvetskaya
KU Leuven, Belgium
yuliya.miadzvetskaya@kuleuven.be

Guillaume Mockly
Trialog, Paris, France
guillaume.mockly@trialog.com

Edgardo Montes de Oca
Montimage, Paris, France
edgardo.montesdeoca@montimage.com

Ana M. Moreno
UPM, Spain

Rubén Mulero
TECNALIA, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastián, Spain

Victor Muntés-Mulero
Beawre, Barcelona, Spain
victor.muntes-mulero@beawre.com

Luong Nguyen
Montimage, Paris, France
luong.nguyen@montimage.com

Phu Nguyen
SINTEF Digital, Oslo, Norway
phu.nguyen@sintef.no

Sarah Noyé
TECNALIA, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastián, Spain

Daniel Pakkala
VTT, Finland

Alexander Palm
paluno – The Ruhr Institute for Software Technology University of Duisburg-Essen, Essen, Germany
alexander.palm@paluno.uni-due.de

Francisco Parrilla
Indra Sistemas S.A, Madrid, Spain
fparrilla@indra.es

Sheila Puente
TECNALIA, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastián, Spain

Jukka Rantala
Nokia, Finland

Angel Rego
TECNALIA, Basque Research and Technology Alliance (BRTA), Derio, Spain
angel.rego@tecnalia.com

Erkuden Rios
TECNALIA, Basque Research and Technology Alliance (BRTA), Derio, Spain
erkuden.rios@tecnalia.com

Valère Robin
Orange, France

Gérald Rocher
Université Côte d'Azur, I3S/CNRS Sparks, Sophia Antipolis, France
gerald.rocher@univ-cotedazur.fr

Jukka Saarinen
Nokia, Finland

Arezki Slimani
Evidian – ATOS, France
arezki.slimani@evidian.com

Arnor Solberg
Tellu, Asker, Norway
arnor.solberg@tellu.no

Cristóbal Costa Soria
ITI, Spain

Bjørn Skjellaug
SINTEF Digital, Oslo, Norway

Hui Song
SINTEF Digital, Oslo, Norway
hui.song@sintef.no

Mike Surridge
IT Innovation, UK

Tuomo Tuikka
VTT, Finland

Jean-Yves Tigli
Université Côte d'Azur, I3S/CNRS Sparks, Sophia Antipolis, France
jean-yves.tigli@univ-cotedazur.fr

Josef Urban
Nokia Bell Labs, Germany

Thorsten Weyer
paluno – The Ruhr Institute for Software Technology University of Duisburg-Essen, Essen, Germany

Oscar Zanutto
ISRAA, Italy
faber@israa.it