Development and Operation of Trustworthy Smart IoT Systems: The ENACT Framework

Nicolas Ferry[1], Jacek Dominiak[4], Anne Gallon[3], Elena González[4], Eider Iturbe[6], Stéphane Lavirotte[2], Saturnino Martinez[6], Andreas Metzger[5], Victor Muntés-Mulero[4], Phu H. Nguyen[1], Alexander Palm[5], Angel Rego[6], Erkuden Rios[6], Diego Riviera[7], Arnor Solberg[8], Hui Song[1], Jean-Yves Tigli[2], and Thierry Winter[3]

[1] SINTEF Digital, Oslo, Norway,
name.surname@sintef.no
[2] Université Côte d'Azur, CNRS, I3S, France
name.surname@unice.fr
[3] EVIDIAN, Les Clayes-sous-Bois, France
name.surname@evidian.com
[4] Beawre, Barcelona, Spain
name.surname@beawre.com
[5] Paluno (The Ruhr Institute for Software Technology), University of Duisburg-Essen, Germany
name.surname@paluno.uni-due.de
[6] Fundación Tecnalia Research & Innovation, Derio, Spain.
name.surname@tecnalia.com
[7] Montimage, Paris, France.
name.surname@montimage.com
[8] TellU AS, Oslo, Norway.
name.surname@tellu.no

Abstract To unleash the full potential of IoT, it is critical to facilitate the creation and operation of trustworthy Smart IoT Systems (SIS). Software development and delivery of SIS would greatly benefit from DevOps as devices and IoT services requirements for reliability, quality, security and safety are paramount. However, DevOps practices are far from widely adopted in the IoT, in particular, due to a lack of key enabling tools. In last year paper at DevOps'18, we presented the ENACT research roadmap that identified the critical challenges to enable DevOps in the realm of trustworthy SIS. In this paper, we present the ENACT DevOps Framework as our current realization of these methods and tools.

Keywords: DevOps, Internet-of-Things, Trustworthiness

1 Introduction

To fully realize the potential of the IoT, it is important to facilitate the creation and operation of the next generation IoT systems that we denote as Smart IoT Systems (SIS). SIS typically need to perform distributed processing and coordinated behaviour across IoT, edge and cloud infrastructures, manage the closed loop from sensing to actuation, and cope with vast heterogeneity, scalability and dynamicity of IoT systems and their environments.

Major challenges are to improve the efficiency and the collaboration between operation and development teams for the rapid and agile design and evolution of the system. To address these challenges, the ENACT H2020 project [7] embraces the DevOps approach and principles. DevOps [10] has recently emerged as a software development practice that encourages developers to continuously patch, update, or bring new features to the system under operation without sacrificing quality. Software development and delivery of SIS would greatly benefit from DevOps as devices and IoT services requirements for reliability, quality, security and safety are paramount. However, even if DevOps is not bound to any application domain, many challenges appear when the IoT intersects with DevOps. As a result, DevOps practices are far from widely adopted in the IoT, in particular, due to a lack of key enabling tools [19,13].

Current DevOps solutions typically lack mechanisms for continuous quality assurance [13], *e.g.*, mechanisms to ensure end-to-end security and privacy as well as mechanisms able to take into consideration open context and actuation conflicts (*e.g.*, allowing continuous testing of IoT systems within emulated and simulated infrastructures). It also remains challenging to perform continuous deployment and evolution of IoT systems across IoT, edge, and cloud spaces [13]. Our recent systematic studies have found a lack of addressing trustworthiness aspects in the current IoT deployment and orchestration approaches [15,14]. These are key features to provide DevOps for trustworthy SIS.

To address this issue, ENACT will deliver a set of tools for the DevOps of trustworthy SIS. In our former paper [7], we presented the ENACT research roadmap that identified the critical challenges to enable DevOps in the realm of trustworthy SIS. We also introduced the related contribution of the ENACT project and an evolution of the DevOps methods and tools to address these challenges. In this paper, we aim at presenting the ENACT DevOps Framework as our current realization of these methods and tools.

The remainder of this paper is organized as follows. Section 2 presents the overall architecture of the ENACT DevOps Framework, including its architecture and details about the different tools that form this framework. Section 3 exemplifies how they can be used all together to develop and operate trustworthy SIS. Section 4 details how trustworthiness is used as a driver for feedback between Ops and Dev activities. Section 5 summarizes the list of models shared between all the ENACT tools. Finally, Section 6 presents related works and Section 7 concludes.

2 The ENACT Approach

The ENACT DevOps approach is to evolve 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 [9]).

2.1 Conceptual Architecture of the ENACT DevOps Framework

ENACT provides an integrated DevOps Framework composed of a set of loosely coupled tools. Still, these tools can be seamlessly combined, and they can easily integrate with

existing IoT platform services and enablers. Figure 1 shows the set of tools that forms the ENACT DevOps Framework as well as the relationships between these tools. This conceptual architecture consists of five layers, where each layer denotes a particular level of abstraction, complexity and dynamic.

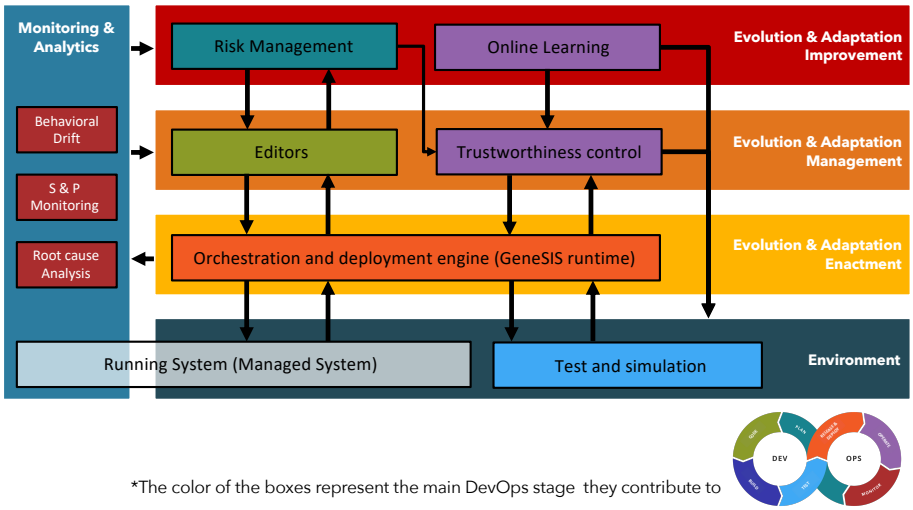

Figure 1. The ENACT Overall Architecture

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 tool 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. Risk management tools typically relates to the Plan stage in the DevOps process. However, in ENACT we will extend the scope to provide continuous risk management. On the other hand, the Online Learning tool 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 tools 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).

2.2 Evolution & Adaptation Improvement Layer

The improvement layer consists of two tools: *(i)* the Risk Management tool and *(ii)* the Online Learning tool.

Risk Management: The Risk Management tool 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. The approach is an evolution of the MUSA Risks management tool [17] that focused security for cloud-based systems. The extension comes with the ability to define IoT-related risks, both by selecting predefined risks stored in a catalogue or allowing users to define them. It also allows for the assessment of such risks. The Risk Management tool integrates with the DevOps cycle to continuously monitor the risk mitigation status though evidences collectors and, thus, enable continuous risk management. The Risk Management tool consumes as input a catalogue of risk treatments, catalogues of security and privacy controls, and an orchestration and deployment model. It produces as output a risk management plan, which includes a set of risk treatment suggestions, and continuous information about the status of the implementation of the different treatments, as well as the effectiveness of these treatments when this information is available.

Online Learning: The Online Learning tool supports a system in the way it adapts itself. Adaptation helps a system to maintain its quality requirements in the presence of environment changes. To develop an adaptive system, developers need an intricate understanding of the system implementation and its environment, and how adaptation impacts system quality. However, due to design-time uncertainty, anticipating all potential

environment changes at design-time is in most cases infeasible. Online learning facilitates addressing design-time uncertainty. By observing the system and its environment at run-time, online learning can automatically refine a system's adaptation capabilities. One of the most widely used online learning techniques is reinforcement learning, which can learn the effectiveness of adaptation actions through interactions with the system's environment. All existing reinforcement learning approaches use value-based reinforcement learning, which are not able to cope with large, continuous environment states (cf. Section 6). To address this issue we instead realize our Online Learning tool, we employ policy-based reinforcement learning, a fundamentally different reinforcement learning technique. In a further state the tool should be able to take behavioural drift information as an input to trigger new learning phases. At this state the online learning tool consumes information about the current environment state of the system to adapt, and attributes of the system that can be used to compute a reward to evaluate the current parameter setting. It then produces a new parameter setting which can be applied to the system, so that a new time-step in the underlying sequential decision problem is reached.

2.3 Evolution & Adaptation Management Layer

The evolution and adaptation management layer is composed of two main groups of tools: *(i)* the editors and *(ii)* the trustworthiness controls.

The editors are meant to support DevOps engineers in specifying the behaviour and deployment of SIS. This includes:

ThingML: ThingML [12] 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) and 10 different communication protocols. ThingML models can be platform specific, meaning that they can only be used to generate code for a specific platform (for instance to exploit some specificities of the platform); or they can be platform independent, meaning that they can be used to generate code in different languages. In ENACT, ThingML can be used to specify the behaviour of software components that will be part of a SIS. As part of ENACT, ThingML is extended with mechanisms to monitor and debug the execution flow of a ThingML program. Following the ThingML philosophy, the proposed monitoring mechanism is platform independent, meaning that the concepts monitored at the target program execution are refined as ThingML concepts. The ThingML run-time consumes as input ThingML programs and produces as output an implementation of an application component.

GeneSIS: GeneSIS [6] is a tool to support the continuous orchestration and deployment of SIS, allowing decentralized processing across heterogeneous IoT, edge, and cloud infrastructures. GeneSIS includes: *(i)* a domain-specific modelling language to model the orchestration and deployment of SIS; and *(ii)* an execution engine that supports the orchestration of IoT, edge, and cloud services as well as their automatic deployment

across IoT, edge, and cloud infrastructure resources. GeneSIS is being built as part of ENACT and inspires from CloudML [4], a tool for the deployment of multi-cloud systems. GeneSIS will also embed the necessary concepts (both in the language and in the execution engine) to support the deployment of security and privacy controls [5] and monitoring mechanisms as well as for the deployment of actuation conflict managers. Finally, GeneSIS will offer specific mechanisms to support the deployment of ThingML programs. The GeneSIS execution engine consumes as inputs deployable artefacts (*i.e.*, implementation of application components that need to be allocated on host services and infrastructure) and a GeneSIS deployment model. It produces as output a GeneSIS deployment model with run-time information (*e.g.*, IP addresses), notifications about the status of a deployed system, and actually deploys the SIS.

Actuation Conflict Manager: The Actuation Conflict Manager tool supports the identification, analysis and resolution of actuation conflicts. The identification of actuation conflicts is done during development and thus relies on the overall architecture of the SIS. It consists in identifying concurrent accesses to the same actuator or actuators interacting through a shared physical environment. The analysis of the conflicts consists in understanding the flow of data coming to the actuators. This includes understanding where the data originated from as well as the path it followed (*i.e.*, through which components) before reaching the actuator. The conflict resolution will provide DevOps engineers with the ability to either *(i)* select an off-the-shelf actuation conflict manager or *(ii)* design their own actuation conflict manager with safety requirements. Finally, the actuation conflict manager tool will support the integration of the actuation conflict manager into the SIS. The Actuation Conflict Manager consumes, as input, an orchestration and deployment model and produces and provides, as output, an actuation conflict manager together with a new orchestration and deployment model (that includes the actuation conflict manager).

The trustworthiness control tools are meant to ensure the trustworthiness of a SIS. This includes:

Diversifier: At development time, the Diversifier consumes as input a GeneSIS deployment model or a ThingML behaviour specification and produces as output multiple diverse specifications. At the current stage, the architecture diversification is focused on diversifying the composition of reusable blocks, and the code (behaviour) diversifier is focused on the diversification of communication protocols. At run-time, the diversifier aims at managing a large and dynamic number of sub-systems with emerging and injected diversity, in order to achieve the robustness and resilience of the entire system. In short, it monitors and records the diversity among subsystems, managing the lifecycles of these subsystems, and controls the upgrading, deployment and modification of software components on these subsystems.

Security and Privacy controls: Security and Privacy control tool is a set of multiple mechanisms that, in a complementary way, can provide security and/or privacy to different elements of a SIS. It will provide security controls embedded in the IoT platform related to integrity, non-repudiation and access control. Moreover, the communications

sent through the IoT platform can be stopped based on specific pre-defined rules related to the behaviour of the SIS. The tool will be further enhanced with a Security and Privacy Control Manager that can enable, disable and configure the controls provided within the Security and Privacy control tool.

Context-Aware Access Control: The Context-Aware Access Control tool is a solution for dynamic authorization based on context for both IT and OT (operational technologies) domains. In particular, this tool provides Context-aware risk and trust-based dynamic authorization mechanisms ensuring *(i)* that an authenticated IoT node accesses only what it is authorized to and *(ii)* that an IoT node can only be accessed by authorized software components. Access authorizations will be adapted according to contextual information. Context may be for instance the date and time an access authorization is requested, the geolocation of this request, or it can be dynamic attributes coming from other external sources (sensors, other applications, etc.). The Context-Aware Access Control tool consumes as inputs rules for contextual adaptation and access control policies.

2.4 Evolution & Adaptation Enactment Layer

The adaptation enactment layer basically consists of the GeneSIS execution environment. From a deployment model specified using the GENESIS Modelling language, the GENESIS execution environment is responsible for: *(i)* deploying the software components, *(ii)* ensuring communication between them, *(iii)* provisioning cloud resources, and *(iv)* monitoring the status of the deployment. The GENESIS deployment engine implements the Models@Run-time pattern [2] to support the dynamic adaptation of a deployment with minimal impact on the running system. It provides the other tools with interface to dynamically adapt the orchestration and deployment of a SIS.

2.5 System Layer

In addition to the running system, this layer encompasses the test and simulation tool.

Test and simulation: Test and simulation tool provides concepts and tools for running application scenarios against the set of programmed circumstances. The tool-set is aiming to provide a baseline for performance, resilience testing as well as risk management testing. It does replicate the behaviour of previously observed devices and is able to play back the sensors data against the programmed scenarios. The tool consumes the treatments from the risk management group and produces the report for the outputs of the scenarios.

2.6 Monitoring & Analytics Layer

The monitoring and analytics layer is composed of three tools: *(i)* the Security and Privacy Monitoring tool, *(ii)* the Root Cause Analysis tool, and *(iii)* the Behavioural Drift Analysis tool.

Security and Privacy Monitoring: The Security and Privacy Monitoring tool allows the IoT application operator to monitor the security and privacy status of the IoT system at different layers. The tool will capture and analyse data from multiple and heterogeneous sources such as raw data from network, system and application layers, as well as events from security monitoring components such as intrusion prevention and intrusion detection systems (IPS/IDS). All the data will be processed and displayed in a common dashboard, which includes processed information in the form of alerts, statistics and graphs. This information will be further enhanced with a standardized classification of security events (such as MITRE ATT&CK) and candidate security controls that can mitigate the detected potential threat.

Root Cause Analysis: The main objective of the Root Cause Analysis (RCA) module is to provide the ENACT platform with a reliable tool capable of detecting the origin of failures on the system. This engine relies on both instrumentation of the software (logs generation or specific instrumentation software) and monitoring of the devices and network. The RCA tool will use these data as the principal input to generate a graph of the system, which will be used to identify the scope of the detected anomalies. Later, the graph that represents the potential impact of the anomalies will be matched against a previously-assessed anomalies database, whose Root Cause is already known. This process will be further enhanced with feedback from the system administrator, who will be able to decide which is the best match for the anomaly detected. The feedback received from the user (the correct and erroneous matches) will be used by the RCA module to adjust its calculation and learn from the context of the system.

Behavioural Drift Analysis: The objective of the Behavioural Drift Analysis tool is to detect whenever a SIS derives (during operation) from its expected behaviour (at development time) and to provide the DevOps engineer with comprehensible representations and models of the drifting behaviour. The drift is measured via a set of probes and sensors that monitors the behaviour of the SIS and by comparing what is observed to the behaviour modelled during development. The DevOps engineer has thus access to a dashboard with representations that aim at facilitating drifts diagnostic by displaying metrics and drifting behaviour models. The Behavioural Drift Analysis tool consumes, as input, a behavioural model of the SIS as well as implementations of the monitoring probes. It produces, as output, measurements describing behavioural drifts as well as a behavioural model updated based on run-time observations.

3 An Example of the ENACT Workflow

Figure 2 depicts an example of workflow between the ENACT development tools.

First, a DevOps engineer can use GeneSIS (aka., the orchestration and deployment tool) to specify the overall architecture of a SIS (①). This model can thus serve as input for the Risk Management tool, which will help conducting a risk analysis and assessment and may result in a set of mitigation actions, for instance advocating the use of a specific set of security mechanisms (②). As a result, the DevOps engineer may update the model describing the architecture of the SIS before its refinement into a

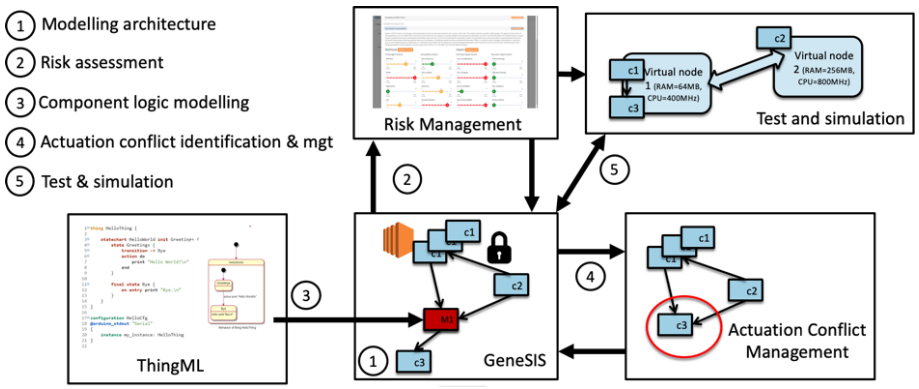

Figure 2. The ENACT Development Toolkit

proper deployment model. The DevOps engineer might also use ThingML to implement some of the software components that should be deployed as part of the SIS (③). At this stage, the Actuation Conflict Manager enabler can be used to identify actuation conflicts – *e.g.*, concurrent accesses to an actuator (④). This enabler will support the DevOps engineer in either selecting or designing an actuation conflict manager to be deployed as part of the SIS (typically as a proxy managing the accesses to the actuator). Finally, the SIS can be simulated and tested, in particular against security threats and scalability issues (⑤) before being sent to GeneSIS for deployment.

As depicted in Figure 3, before deployment, the DevOps engineer may use the Diversifier to increase the overall robustness of the system (⑥). This tool can generate different variants, but still functionally equivalent, of the components of the SIS (*e.g.*, different versions of the software components are deployed making the overall SIS more robust to security and privacy attacks). After diversification, the SIS can be deployed using GeneSIS. Once the SIS in operation, a set of ENACT tools are responsible for its run-time monitoring and control. Monitoring tools are depicted on the right part of Figure 3 (⑦). First, the Behavioural Drift Analysis tool can be used to understand to which extent a SIS behaves (in the real world) as expected (during development). This is important as a SIS typically operates in the midst of the unpredictable physical world and thus all the situations it may face at run-time may not have been fully understood or anticipated during development. Second, the Security and Privacy monitoring tool can be used to identify security and privacy breaches. Third, in case of failure, the Root Cause Analysis tool provides DevOps teams with insights on the origin of that failure. Control tools are depicted on the left part of Figure 3. First, the Context-aware Access Control tool can be used to manage accesses from services to sensors and actuators and the other way around (⑧). Accesses can be granted or removed based on context information. Finally, the Online Learning enabler uses reinforcement learning techniques to enhance the adaptation logic embedded into a SIS (⑨).

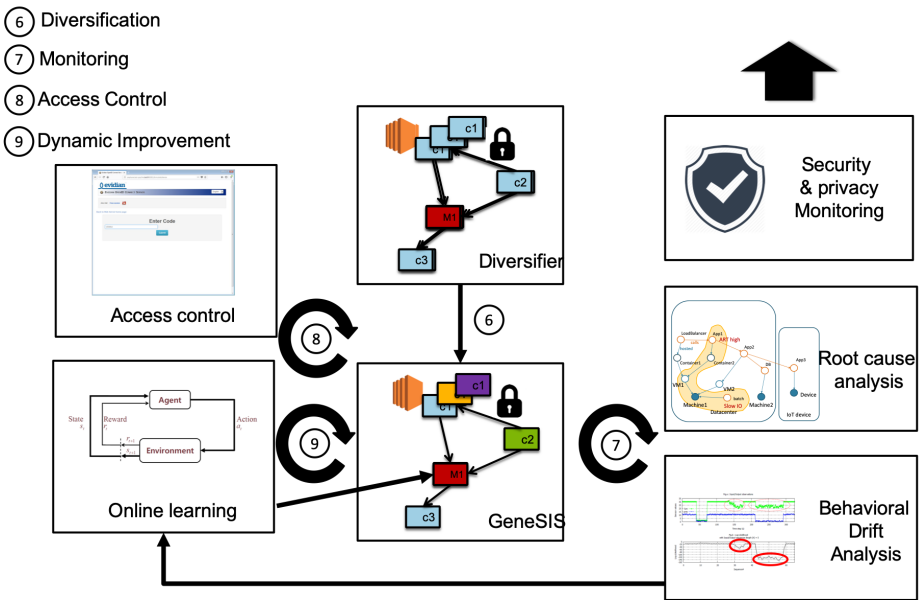

Figure 3. The ENACT Operation Toolkit

4 Trustworthiness as a Driver for Feedback Between Ops and Dev

One of the core values of DevOps is to improve synergies and communications between development and operation activities. When software is in production it often produces a plethora of data that ranges from logs to high level indicators (*e.g.*, performance indicators, context monitoring). All these data gathered at run-time can serve as valuable feedback [3] from operation to development and, in particular, it can serve to evolve and improve the SIS by triggering a new development cycle. Some of the challenges are thus to properly integrate development and operation tools and to seamlessly integrate feedbacks from run-time into development tools. In this section, we illustrate how we address this challenge in the project.

In the context of ENACT, one of the main drivers for triggering a new development cycle of a SIS on the basis of observations from operation is to improve its trustworthiness, and in particular its security, privacy, resilience, reliability and safety. In the following (see Figure 4) we illustrate how security and privacy run-time information can be used to continuously assess risk and can, in turn, lead to an evolution of a SIS.

First, a DevOps engineer can use GeneSIS (aka., the orchestration and deployment tool) to specify the overall architecture of a SIS. The resulting deployment model can be sent to the Risk Management enabler. The latter is thus used to perform a risk assessment that will result in a set of risk treatment suggestions. A risk treatment can be a procedure to follow in order to mitigate a risk or simply a recommendation to use a specific software solution or mechanism (*e.g.*, a security control solution such as the Context-Aware Access Control). It is worth noting that a procedure can also in-

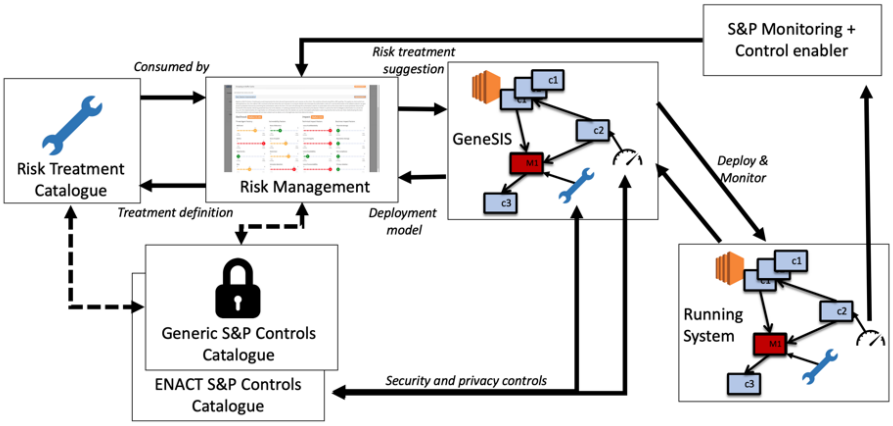

Figure 4. Continuous Risk Management

clude a recommendation for using a specific software solution or mechanism. These recommendations are typically linked to concrete implementations of the solution or mechanisms. In particular, it can leverage a generic security and privacy controls catalogue that includes state of the art security and privacy solutions or the ENACT security and privacy controls catalogue (*i.e.*, the controls implemented in ENACT). It is worth noting that by security and privacy controls we not only refer to mechanisms to implement security or privacy measures but also to mechanisms to monitor security and privacy. The Risk Management enabler embeds a catalogue of risk treatments and can be used to specify and add new ones into the catalogue. From the risk treatment suggestions, the DevOps engineer may decide to evolve its deployment model specifying that specific security and privacy controls (whose implementation is indicated in the suggestion and depicted in Figure 4 as the green icons) should be deployed together with the SIS. After deployment, GeneSIS monitors the status of the deployment whilst the Security and Privacy controls enablers gather security and privacy data from the probes deployed together with the systems. Both tools send some of the gathered metrics to the Risk Management enablers. These metrics are associated to the risk models and used to continuously assess risk.

In the following (see Figure 5), we illustrate how run-time information from the root cause analysis and behavioural drift analysis enablers can be used to improve the resilience, reliability and safety of a SIS.

The Behavioural drift analysis tool aims at observing the actual behaviour of a SIS at run-time and at comparing it with the behaviour that was expected at development time. One result of the comparison is a value called: behavioural drift metric. A behavioural drift may result from a problem in the conception of the SIS, an indirect actuation conflict (*e.g.*, applications are properly design but their actions on the physical environment are somehow conflicting) or an unexpected reaction of the surrounding physical environment. Because it is difficult for a DevOps engineer to understand and take ac-

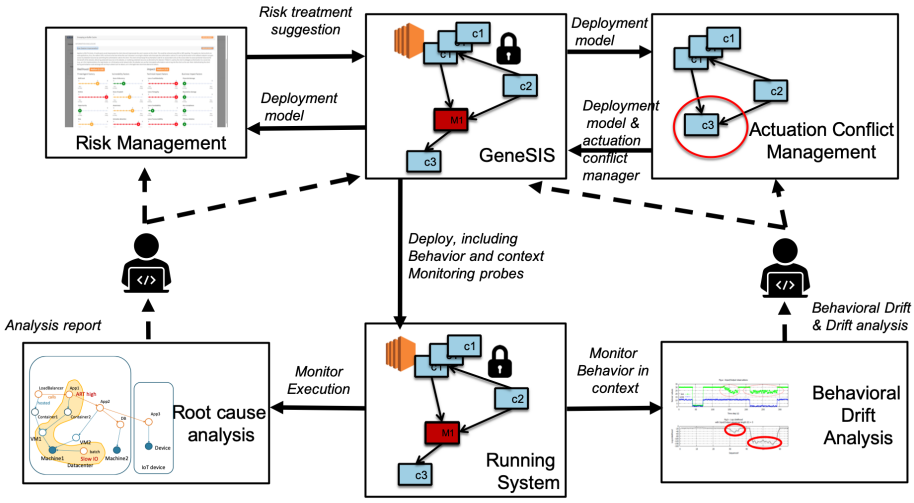

Figure 5. Continuous Improvement of Resilience, Reliability, and Safety

tions simply on the basis of a behavioural drift metric, the tool performs an analysis of this drift, which consists in comparing the model of the expected system's behaviour and the observed one on the basis of what is actually happening at run-time (*e.g.*, the behaviour model is no more deterministic, some probabilities of transitions between states can increase from zero and others can decrease from one). In case a behavioural drift is observed, the tool will provide the DevOps engineer with the analysis, the latter can in turn adapt the SIS via GeneSIS.

The Root cause analysis tool will observe the execution of the system and, in case of failures, report on its origin. Such report will be provided to the DevOps engineer, who can in turn adapt the SIS via GeneSIS or perform a new risk assessment.

5 Shared Models and Artefacts

The ENACT Framework tools manipulate and exchange different types of models and software artefacts as illustrated in Figure 6. Models are represented as rectangles whilst software artefacts (*i.e.*, binaries, scripts, etc. which are generated from or are part of a tool, and used or managed by another) are depicted by circles. A model is embedded in a tool when it is directly used in the internal of the tool.

It is worth noting that the GeneSIS and ThingML models are the most reused amongst the tools. This is because they are the key models specifying the architecture and behavior of a SIS whilst the other are mainly used to manage trustworthiness aspects of the SIS. In the following we shortly describe each model:

- **GeneSIS deployment Model:** is written in a domain-specific modelling language to specify deployment model – *i.e.*, the orchestration and deployment of SIS across the IoT, edge, and cloud spaces.

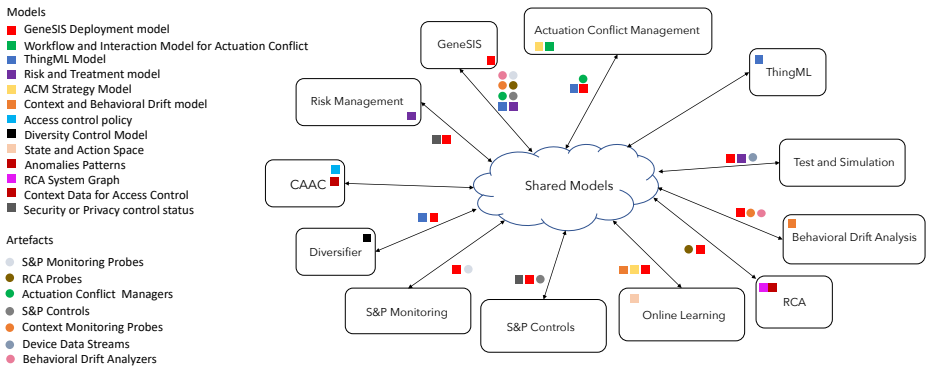

Figure 6. Models Manipulated and Exchanged in the ENACT Framework

- **ThingML Model:** is written in a domain specific modelling language to specify the behavior of distributed software components.
- **Workflow and Interaction Model for Actuation Conflict Model:** describes the interactions between the software components that form a SIS and their relationship with actuators, thus, supporting the identification of both direct and indirect actuation conflict.
- **Risk and Treatment Model:** describes risk and treatment suggestions.
- **ACM Strategy:** takes the form of a set of extended ECA rules describing the behavior of a custom actuation conflict manager.
- **Context and Behavioural Drift:** can be used to describe the SIS operational context and to describe its observed behavior (compared to the expected one).
- **Access Control Policy:** is a set of rules that define whether a user or device must be permitted or denied accessing to a resource.
- **Context Data for Access Control:** provides contextual data on a user and his devices. These data are dynamic attributes and come from other external sources.
- **Diversity Control Model:** a model maintained at run-time that reflects the status in term of deployment, software version and health of a fleet of IoT systems or sub-systems.
- **State and Action Space:** for the online learning tool, the state space represents different environment situations, while the action space represents the different actions the online learning tool may execute to improve the adaptation logic of the Smart IoT system.
- **Anomalies Pattern:** is used to describe a set of patterns, which will be the baselines for the RCA enabler to check the existence of anomalies and maps them to their Root Cause.
- **RCA System Graph:** is a snapshot of the current status (*e.g.*, network activity, system logs, detected anomalies) of the monitored system.
- **Security and Privacy Control Status:** provides information about the status of the security and privacy mechanisms used in the SIS. It is used to understand the current status of the risk but also the progression of the treatment.

In the following we shortly describes each software artefact:

- **Security and Privacy Monitoring Probes:** are deployed together with the SIS with the aim to monitor the status of specific security and privacy aspects. There are probes at the network, application, and system levels.
- **Root Cause Analysis Probes:** retrieve information from the logs generated by the devices on the IoT System, but they will also be able to interact with hardware-based probes (such as the MMT-IoT Sniffer) and software-based solutions (such as Snort and Suricata).
- **Actuation Conflict Managers:** are deployed together with the SIS and are responsible for managing the accesses to conflicting actuators (direct or indirect). All accesses to the actuators should go via the actuation conflict manager. Actuation conflict managers are either provided off-the-shelf or can be designed for a specific type of conflict.
- **Security and Privacy Controls:** are implementations of security and privacy mechanisms to be deployed together with the SIS.
- **Context Monitoring Probes:** are monitoring the context of a SIS. They are deployed together with the SIS and are, in particular, used by the behavioral drift analyzers.
- **Device Data Streams:** are records of devices outputs and inputs used by the test and simulation tool to replay devices behavior.
- **Behavioral Drift Analyzers:** are software components generated by the Behavioral drift analysis tool. They are responsible for analysing a specific behavior of a SIS. Multiple Behavioral Drift Analyzers can be deployed together with the SIS.

6 Related Work

For some years now, multiple tools and solutions have emerged to support the DevOps of software systems and in particular to automate their testing, build, deployment and monitoring. However, to the best of our knowledge, there is no DevOps support tailored for smart IoT systems today [19,13]. According to [19] a key reason is: "the extremely dynamic nature of IoT systems poses additional challenges, for instance, continuous debugging and testing of IoT systems can be very challenging because of the large number of devices, dynamic topologies, unreliable connectivity, and heterogeneous and sometimes invisible nature of the devices". In the following we discuss related work for both the development and operation of SIS.

Continuous development of SIS: The survey in [14] illustrates a lack of approaches and tools specifically designed for supporting the continuous deployment of software systems over IoT, edge, and cloud infrastructure. For example, several solutions are available on the market for the deployment of cloud-based systems such as CloudMF [4], OpenTOSCA [18], Cloudify[9], and Brooklyn[10]. Those are tailored to provision and

[9] http://cloudify.co/

[10] https://brooklyn.apache.org

manage virtual machines or PaaS solutions. In addition, similar tools focus on the management and orchestration of containers, *e.g.*, Docker Compose[11], Kubernetes[12]. Opposed to hypervisor virtual machines, containers such as Docker containers leverage lightweight virtualization technology, which executes directly on the operating system of the host. As a result, Docker shares and exploits a lot of the resources offered by the operating system thus reducing containers' footprint. Thanks to these characteristics, container technologies are not only relevant for cloud infrastructure but can also be used on edge devices. On the other side, few tools such as Resin.io and ioFog 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 for the software component to be deployed 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.*, ARMv6 for a Raspberry Pi) and a Docker image is created before being deployed on the target hosting device(s). However, Resin.io offers limited support for the deployment and management of software components on tiny devices that cannot host containers. The same applies to Microsoft IoT Hub[13].

In addition, the survey in [14] also highlights that very few primary IoT deployment studies address *(i)* security and privacy aspects, and *(ii)* the management of actuators (and actuation conflict). Even if no DevOps solutions for IoT systems embed specific mechanisms for the management of actuation conflicts, the core of this challenge relates to the generic problem of managing features interactions. Indeed, when a global functionality is obtained from a set of shared features, there is a risk for unintended and undesirable interactions between the features. However, because current work on this topic do not focus on the IoT application domain, they encompass the following weaknesses. They give a low degree of importance to *(i)* the modelling of the physical environment as part of the conflicts identification process, and *(ii)* to reusability, scalability and dynamicity as part of the resolution process.

Similarly, there is a lack of risk analysis methodologies that are adapted to agile contexts but still achieve the level of analysis and detail provided by traditional risk assessment and mitigation techniques, in particular related to NFRs. Fitzgerald *et al* [8] illustrate how Lean Thinking [20] can be applied to continuous software engineering. Authors even go beyond software development and consider issues such as continuous use, continuous trust, etc., coining the term "Continuous" (Continuous Star). However, they do not explicitly tackle challenges related to continuous risk management.

Continuous operation of SIS: By observing the system and its environment at runtime, online learning can automatically refine a system's adaptation capabilities. One of the most widely used online learning techniques is reinforcement learning, which can learn the effectiveness of adaptation actions through interactions with the system's

[11] https://docs.docker.com/compose/

[12] https://kubernetes.io

[13] https://azure.microsoft.com/fr-fr/services/iot-hub/

environment. However, so far, all existing reinforcement learning approaches use value-based reinforcement learning and thus face two key limitations. First, they face the exploration/exploitation dilemma, which requires developers to fine-tune the amount of exploration to ensure convergence of the learning process. Second, most approaches store the learned knowledge in a lookup table, which requires developers to manually quantify environment states to facilitate scalability. To realize the Inline Learning too, we thus automate both these manual activities by employing policy-based reinforcement learning, a fundamentally different reinforcement learning technique. Thereby, our Online Learning tool is able to cope with large, continuous environment states. In a further state the tool should be able to take behavioural drift information as an input to trigger new learning phases.

Online learning is meant to be performed at run-time, considering observations about the physical environment, the state of the system, and so the context. Context-awareness is key to collect sensor data, to understand it and to provide valuable information to reasoning engines. Since the first definition of context [1] a lot of middleware and software frameworks have emerged. Already in 2014, [16] finds 50 context-aware solutions in the scientific literature and today lot of well-known approaches are available [11] to collect sensors and probes data leveraged for modelling various contextual concerns (location, situation, social environment, etc.). However, SIS pose new challenges. Indeed, as far as physical things are concerned, no guaranty can be made on their availability on the long run. The underlying infrastructure of SIS can thus be volatile. Moreover, the purpose of some of these systems can only be achieved from interactions with the physical environment through actuators (*e.g.*, Heating, Ventilation and Air-Conditioning controllers). In this context, these systems can possibly be affected by unanticipated physical processes over which they have no control, leading their behaviour to potentially drift over time in the best case or to malfunction in the worst case. As said, many platforms include context awareness and monitoring mechanisms (*e.g.*, SOFIA2[14], FIWARE[15] with the Orion Context Broker for instance). However, these platforms do not consider behavioural drift monitoring as an awareness criterion. This is what is addressed by our behavioral drift analysis tool.

7 Conclusion

We presented the ENACT DevOps Framework which offers a set of novel solutions to address challenges related to the development, operation, and quality assurance of trustworthy smart IoT systems that need to be distributed across IoT, edge and cloud infrastructures and involve both sensors and actuators. These enablers are under development as part of the ENACT H2020 project and will be delivered as open source artefacts.

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

[14] https://sofia2.com
[15] https://www.fiware.org

References

1. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: International symposium on handheld and ubiquitous computing. pp. 304–307. Springer (1999)
2. Blair, G., Bencomo, N., France, R.: Models@run.time. IEEE Computer **42**(10), 22–27 (2009). https://doi.org/10.1109/MC.2009.326
3. Cito, J., Wettinger, J., Lwakatare, L.E., Borg, M., Li, F.: Feedback from operations to software development—a devops perspective on runtime metrics and logs. In: International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment. pp. 184–195. Springer (2018)
4. Ferry, N., Chauvel, F., Song, H., Rossini, A., Lushpenko, M., Solberg, A.: Cloudmf: Model-driven management of multi-cloud applications. ACM Transactions on Internet Technology (TOIT) **18**(2), 16 (2018)
5. Ferry, N., Nguyen, P.H.: Towards model-based continuous deployment of secure IoT systems. In: 1st International Workshop on DevOps@MODELS (2019)
6. Ferry, N., Nguyen, P.H., Song, H., Novac, P.E., Lavirotte, S., Tigli, J.Y., Solberg, A.: Genesis: Continuous orchestration and deployment of smart IoT systems. In: 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC). vol. 1, pp. 870–875 (Jul 2019). https://doi.org/10.1109/COMPSAC.2019.00127
7. Ferry, N., Solberg, A., Song, H., Lavirotte, S., Tigli, J.Y., Winter, T., Muntés-Mulero, V., Metzger, A., Velasco, E.R., Aguirre, A.C.: 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. pp. 112–127. Springer (2018)
8. Fitzgerald, B., Stol, K., O'Sullivan, R., O'Brien, D.: Scaling agile methods to regulated environments: An industry case study. In: International Conference on Software Engineering. pp. 863–872. ICSE '13, IEEE Press (2013), http://dl.acm.org/citation.cfm?id=2486788.2486906
9. Griffor, E.R., Greer, C., Wollman, D.A., Burns, M.J.: Framework for cyber-physical systems: Volume 1, overview. Tech. rep. (2017)
10. Humble, J., Farley, D.: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional (2010)
11. Künzler, F., Kramer, J.N., Kowatsch, T.: Efficacy of mobile context-aware notification management systems: A systematic literature review and meta-analysis. In: 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). pp. 131–138. IEEE (2017)
12. Morin, B., Fleurey, F., Husa, K.E., Barais, O.: A generative middleware for heterogeneous and distributed services. In: 19th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE). pp. 107–116. IEEE (2016)
13. NESSI: Software continuum: Recommendations for ict work programme 2018+. Nessi report (2016)
14. Nguyen, P.H., Ferry, N., Erdogan, G., Song, H., Lavirotte, S., Tigli, J.Y., Solberg, A.: Advances in deployment and orchestration approaches for IoT - a systematic review. In: 2019 IEEE International Congress on Internet of Things (ICIOT). pp. 53–60 (July 2019). https://doi.org/10.1109/ICIOT.2019.00021
15. Nguyen, P.H., Ferry, N., Erdogan, G., Song, H., Lavirotte, S., Tigli, J.Y., Solberg, A.: The preliminary results of a mapping study of deployment and orchestration for IoT. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 2040–2043. SAC '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3297280.3297617, http://doi.acm.org/10.1145/3297280.3297617

16. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: A survey. IEEE communications surveys & tutorials **16**(1), 414–454 (2013)
17. Rios, E., Iturbe, E., Larrucea, X., Rak, M., Mallouli, W., Dominiak, J., Muntés, V., Matthews, P., Gonzalez, L.: Service level agreement-based gdpr compliance and security assurance in (multi) cloud-based systems. IET Software (2019)
18. da Silva, A.C.F., Breitenbücher, U., Képes, K., Kopp, O., Leymann, F.: 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. pp. 181–182. ACM (2016)
19. Taivalsaari, A., Mikkonen, T.: A roadmap to the programmable world: software challenges in the iot era. IEEE Software **34**(1), 72–80 (2017)
20. Womack, J., Jones, D.: Lean Thinking: Banish Waste and Create Wealth in Your Corporation, Revised and Updated. Free Press (2003), https://books.google.co.uk/books?id=l8hWAAAAYAAJ