# ENACT: Development, Operation, and Quality Assurance of Trustworthy Smart IoT Systems

Nicolas Ferry[1], Arnor Solberg[1] Hui Song[1], Stéphane Lavirotte[2], Jean-Yves Tigli[2], Thierry Winter[3], Victor Muntés-Mulero[4], Andreas Metzger[5], Erkuden Rios Velasco[6], and Amaia Castelruiz Aguirre[6]

[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
thierry.winter@evidian.com
[4] CA Technologies, Barcelona, Spain
victor.muntes@ca.com
[5] paluno (The Ruhr Institute for Software Technology), University of Duisburg-Essen, Germany
andreas.metzger@paluno.uni-due.de
[6] Fundación Tecnalia Research & Innovation, Derio, Spain.
name.surname@tecnalia.com

**Abstract** To unleash the full potential of IoT and flourishing innovations in application domains such as eHealth or smart city, it is critical to facilitate the creation and operation of trustworthy Smart IoT Systems (SIS). Since SIS typically operate in a changing and often unpredictable environment, the ability of these systems to continuously evolve and adapt to their new environment is decisive to ensure and increase their trustworthiness, quality and user experience. The DevOps movement advocates a set of software engineering best practices and tools, to ensure Quality of Service whilst continuously evolving complex systems. However, there is no complete DevOps support for trustworthy SIS today. In this paper we present a research roadmap to enable DevOps in such systems and introduce the ENACT DevOps concepts and Framework.

**Keywords:** Internet of Things, DevOps, Trustworthiness

## 1 Introduction

By 2020, Gartner envisions that 21 billion Internet-of-Things (IoT) endpoints will be in use[7], representing great business opportunities. However, complex challenges remain to be solved to efficiently exploit the full potential of the rapidly growing IoT infrastructure. Until now, IoT system innovations have been mainly concerned with sensors, device management and connectivity, with the mission to gather data for processing and analysis in the cloud in order to aggregate information and knowledge [1]. This approach has conveyed significant added value in many application domains, however, it

[7] http://www.gartner.com/newsroom/id/3598917

does not unleash the full potential of the IoT[8]. The next generation IoT systems need to perform distributed processing and coordinated behaviour across IoT, edge and cloud infrastructures [2], manage the closed loop from sensing to actuation, and cope with vast heterogeneity, scalability and dynamicity of IoT systems and their environments. Moreover, the functioning and correctness of such systems will be critical, ranging from business critical to safety critical. Thus, aspects related to trustworthiness such as security, privacy, resilience and robustness, are challenging aspects of paramount importance [1]. Therefore, the next generation of IoT systems needs to be trustworthy. In this paper, we will call them trustworthy smart IoT systems, or for short; trustworthy SIS.

To realize the digital society and to flourish innovations, it is critical to facilitate the creation and operation of trustworthy SIS. However, developing and managing the next generation trustworthy SIS that operates in the midst of the unpredictable physical world represents daunting challenges. For example, to ensure that such systems always work within safe operational boundaries [3] (*e.g.*, controlling the impact that actuators have on the physical world) and to manage conflicting actuation requests. Moreover, the ability of these systems to continuously evolve and adapt to their changing environments is decisive to ensure and increase their trustworthiness, quality and user experience.

This is at the core of the DevOps movement, which advocates a set of software engineering best practices and tools, to ensure Quality of Service whilst continuously evolving complex systems and foster agility, rapid innovation cycles, and ease of use [4]. Therefore, DevOps has been widely adopted in the software industry. However, there is no complete DevOps support for trustworthy smart IoT systems today [5,3]. According to [5], a key reason is "*because of the extremely dynamic nature of IoT systems, it 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*". Current DevOps solutions also lack mechanisms for continuous quality assurance [3], for example, 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 [3]. These are key features to provide DevOps for trustworthy SIS.

In this paper we first provide a research roadmap that identifies the critical challenges to enable DevOps in the realm of trustworthy SIS. We discuss the related contribution of the ENACT DevOps Framework and introduce an evolution of the DevOps methods and tools to address these challenges. Finally, we present the initial ENACT DevOps Framework as our current realization of these methods and tools. The proposed framework will be explored and further developed in the newly founded ENACT H2020 project that started in January 2018.

The remainder of the paper is organized as follows. Section 2 presents the research roadmap in terms of technical challenges and state-of-the-art. Section 3 details the over-

[8] https://ec.europa.eu/digital-single-market/en/internet-of-things

all ENACT approach and illustrates how it will help addressing these research challenges. Section 4 describes the set of enablers that forms the core of the ENACT DevOps Framework and Section 5 concludes.

## 2 ENACT Research Roadmap

The key research question is the following: *how can we tame the complexity of developing and operating smart IoT systems, which (i) involve sensors and* ***actuators*** *and (ii) need to be* ***trustworthy****?*. Our approach is largely to evolve DevOps methods and techniques as baseline to address this issue. We thus refine the research question as follows: *how can we apply and evolve the DevOps tools and methods to facilitate the development and operation of trustworthy smart IoT systems?*. The answer to these questions form the core of the ENACT research roadmap presented below.

**Research Challenge 1: Support continuous delivery of trustworthy SIS.**

*Context:* Very little effort has been spent on providing solutions for the delivery and deployment of application across the whole IoT, edge and cloud space. In particular, there is a lack of languages and abstractions that can be used to support the orchestration of software services and their deployment on heterogeneous devices [6].

*State-of-the-art:* In the past years, multiple tools have emerged to support the building as well as the automated and continuous deployment of software systems with a specific focus on cloud infrastructures (*e.g.*, Puppet[9], Chef[10], Brooklyn[11], TOSCA[12], CloudMF [7], etc.). Model-Driven Engineering techniques have also been investigated to design and reconfigure a network of resource-constrained devices. One trend is to provide an architecture-based approach to design and build flexible embedded systems, (*e.g.*, the Koala model [8] and Think [9]). Kevoree [10] relies on models at run-time to support the dynamic adaptation of distributed cloud and cyber physical systems supporting hot deployment of component types. The HEADS FP7 EU project proposes ThingML [6], a language that provides a set of compilers to derive source code (*e.g.*, Java, C/C++) optimised for various IoT platforms.

**Research Challenge 2: Support the agile operation of trustworthy SIS.**

*Context:* The operation of large-scale and highly distributed IoT systems can easily overwhelm operation teams. Major challenges are then to improve their efficiency and the collaboration with developer teams for rapid and agile evolution of the systems. However, there is a lack of mechanisms dedicated to smart IoT systems able to *(i)* monitor their status, *(ii)* indicate when their behaviour is not as expected, *(iii)* identify

[9] https://puppet.com
[10] https://www.chef.io/
[11] https://brooklyn.apache.org/
[12] https://www.oasis-open.org/committees/tosca

the origin of the problem, and *(iv)* automate typical operation activities[5]. Furthermore, because it is impossible to anticipate all the adaptations a system may face when operating in an open context, there is an urgent need for mechanisms that will automatically maintain the adaptation rules of a smart IoT system.

*State-of-the-art:* A self-adaptive system modifies its own structure and behaviour at run-time to respond to its perception of its environment, of itself and of its requirements [11]. Software developers face two important challenges when building self-adaptive systems [12]. On the one hand, it is difficult or may even be infeasible for software developers to exhaustively explore, anticipate or resolve all possible situations that an adaptive system may encounter during its operation. On the other hand, it is difficult for software developers to determine how a modification of the system's structure or behaviour impacts on the satisfaction of the system's requirements. These two challenges make it hard to determine the required set of system configurations a self-adaptive system needs to implement. In addition, these challenges make it hard to define which configurations to choose in response to the anticipated situations. A self-adaptive system thus may encounter situations that have not been fully understood or anticipated in the software development process [13]. Should an unanticipated situation occur at run-time, a self-adaptive system thus may wrongly reconfigure itself, (*e.g.*, the self-adaptive system may choose a configuration that is not able to address the situation). To attack the aforementioned problems, researchers have started applying online learning techniques to improve the way that a self-adaptive system adapts [14,15]. Online learning means that learning is performed at run-time, taking into account observations about the actual system execution and system environment. Online learning incrementally updates the self-adaptive system's knowledge base; *e.g.*, its adaptation rules or the models based on which adaptation decisions are made. However, so far, techniques and methodologies for self-adaptive systems have focused on the system as the entity that may be adapted only. In the presence of IoT actuation, a new avenue for adaptive behaviour becomes possible that has not yet been addressed. In addition, existing online learning techniques for self-adaptive systems do not take into account the impact of system evolution (*i.e.*, the manual modification of the system by humans [16])). During evolution, software developers may modify the system, for instance, to correct bugs, to remove seldom used features, or to introduce new features, thereby also changing the possible adaptations of the system.

Identifying the the root causes of faults or problems is also of major importance. However, performing root-cause analysis in complex and dynamic IoT environments is not deeply studied in the literature. Several IoT-related problems have been detected. For instance, Aggarwal [17] discusses challenges related to incomplete data transmission from sensors for Big Data analytics. Failures in fog computing can be localized at sensor, network, service platform or web application levels [18]. Some other efforts focus on the convergence of Big Data analysis techniques and Cyber Physical Systems [19], describing a data-driven approach to building fault tolerant control systems. However, they use highly accurate mathematical models that do not comply with usual scalability and computational complexity in large IoT structures. Finally, recent work [20] proposes some initial techniques to manage scalability issues for root-cause analysis in IoT and fog environments.

**Research Challenge 3: Support continuous quality assurance strengthening trustworthiness of SIS.**

*Context:* Ensuring quality of service is a complex task that needs to be considered throughout the whole life-cycle of a system. This is all the more exacerbated in the smart IoT system context where it is infeasible for developers and operators to exhaustively explore, anticipate or resolve all possible context situations that a system may encounter during its operation. This is due to the open context in which these systems operate and as a result can hinder their trustworthiness. This is particularly important when the system can have an impact on the physical world through actuators. In addition, testing, ensuring end-to-end security as well as the robustness of such systems is challenging [5]. A major limitation of classical quality assurance tools when applied in IoT is the lack of mechanisms to include, as part of the tests, constraints related to the distribution and infrastructure of IoT and edge computing[13].

*State-of-the-art:* An important aspect of trustworthiness is security and privacy. Different works identify the diverse security and privacy threats of IoT. One of the most prominent is the work of Open Web Application Security Project (OWASP) that identifies the top ten most common vulnerabilities of IoT systems [21] covering the whole IoT architecture layers (from Insecure Web Interface to Poor physical security flaws). Authors in [22] exemplify hands on the "most severe, yet easy to abuse" IoT threats, namely: leakage of the personally identifiable information (PII), leakage of sensitive user information and unauthorized execution of functions. Cvitic et al. [23] analysed the security aspects for each layer of the IoT architecture: the biggest security risk is at perception layer of the IoT architecture due to the specific limitations of devices and the transmission technology used at this layer, followed by the middleware layer based on cloud computing and inherited vulnerabilities of that concept.

Test and validation is also an important aspect of trustworthiness. Testing of IoT Services is currently limited to service functionality by using existing software testing tools (*e.g.*, Katalon Studio[14], HP Unified Functional Testing[15] or CA Technologies Application Test[16]), which are prepared to verify automatically that the service provides an expected answer for a set of specified and known scenarios as well as testing the communication integrity between different devices. Nevertheless, IoT deployment is heavily impacted by the underlying hardware specification, capabilities, availability and changes in the physical environment, typically not considered by current testing tools. The lack of tools for systematically testing IoT services in different large-scale, heterogeneous, physical environments is a research challenge [24]. Real-Time Operating Systems (RTOS) provide a virtual environment in which developers can test their solutions and deploy the software to the physical device which has the same RTOS installed. Several RTOS are available in the market[17], selecting RTOS is largely based on

[13] http://events.windriver.com/wrcd01/wrcm/2016/08/WP-devops-in-the- internet-of-things.pdf
[14] https://www.katalon.com,
[15] https://saas.hpe.com/en-us/software/uft
[16] https://www.ca.com/us/products/ca-application-test.html
[17] FreeRTOS: http://www.freertos.org/index.html, RIoT OS: https://www.riot-os.org/, ERIKA Enterprise: http://erika.tuxfamily.org/drupal/

the set of supported devices. Current combination of RTOS and other IoT middleware are not capable of scaling up to large-scale deployments due to the limited computation resources of centralized emulations. In the literature, there are simulation strategies to approximate IoT behaviour thanks to parallel and distributed computation [25] and multi-level simulation [26]. Several companies started offering their own infrastructure of IoT devices to test applications within a predefined set of popular IoT devices[18]. Although this partially solves the issue of obtaining data from the physical world and provides testing on real devices, availability of these testing services is significantly impacted by the demand for these services and tests are limited to the currently deployed devices and their location with no option to scale or make changes on the devices. Recent efforts focus on producing self-learning IoT virtualizers to facilitate testing in IoT environments[19].

**Research Challenge 4: Leverage the capabilities of existing IoT platforms and fully exploit legacy, proprietary and off-the-shelf software components and devices.**

*Context:* Real IoT systems are never developed from scratch: they build upon off-the-shelf components as well as legacy sub-systems and they can rely on the wide range of IoT platforms that have been developed over the past decade. The large number of IoT platforms available today is not only accidental, it reflects that different IoT systems or even different parts of a single IoT system have different requirements in terms of device management, integration, security, protocols, analytics, visualisation, etc.

*State-of-the-art:* There are hundreds of IoT platforms available[20] among which a vast majority are proprietary and closed solutions. There are also more open solutions such as the European SOFIA[21], FIWARE[22] and CRYSTAL[23] platforms. The SOFIA open source IoT platform was designed to facilitate systems interoperability aiming at promoting new services and applications, specially focusing on smart scenarios. SOFIA is a platform providing physical world information to intelligent services, thus, enabling interoperability among distinct sectors, systems and devices. SOFIA is open-source, multi-platform, multi-language and communication-agnostic. The FIWARE middleware IoT platform is a cloud-based infrastructure that aims at providing a cost-effective creation and delivery of Future Internet applications and services using public APIs to facilitate the application development in many sectors, along with public reference implementations for each component. FIWARE has a dynamic ecosystem providing assets such as a public sandbox for testing purposes or a network of hubs. CRYSTAL defines a Reference Technology Platform that provides a common ground for integrating lifecycle and engineering tools across different engineering disciplines and from multiple

[18] FIT IoT-lab: https://www.iot-lab.info/, IoT lab: http://www.iotinnovationlab.com/, JOSE: https://www.nict.go.jp/en/nrh/nwgn/jose.html, FIESTA-IoT: http://fiesta-iot.eu/

[19] https://knowthings.io

[20] https://iot-analytics.com/current-state-of-iot-platforms-2016/

[21] https://about.sofia2.com/category/idiomas/en/page/3/

[22] https://www.fiware.org/about-us/

[23] http://www.crystal-artemis.eu

stakeholders involved in the development of large scale safety-critical systems. Crystal bases its entire strategy on the use of the emerging open standard OSLC (Open Services for Lifecycle Collaboration) as a standard for the Interoperability Specifications (IOS) in order to achieve common tool and data interoperability in heterogeneous systems engineering development environments.

In the following section we details the overall ENACT approach and contributions to support the DevOps for trustworthy SIS.

## 3 ENACT Approach

The overall ENACT approach is to deliver novel IoT platform enablers to:

1. Enable DevOps in the realm of trustworthy smart IoT systems, and enrich it with novel concepts for end-to-end security and privacy, resilience and robustness strengthening trustworthiness, taking into account the challenges related to "collaborative" actuation and actuation conflicts identification and management.
2. Facilitate the smooth integration of these to leverage DevOps for existing and new IoT platforms and approaches. The ENACT enablers represent novel concepts realised as open software engineering methods and tools.

In the following we summarize the key contribution of the ENACT Framework to the research challenges introduced in Section 2.

*Contribution to Research Challenge 1:* To reduce delivery time and to foster continuous evolution of trustworthy SIS, the ENACT DevOps Framework will provide automation to close the gap between development and operation activities following the philosophy of DevOps [4], and foster the continuous creation, evolution, and deployment of trustworthy SIS. In particular, the ENACT DevOps Framework will deliver enablers to accomplish orchestration of sensors, actuators, software services and topology configurations, and support the automatic testing and deployment of this orchestration across the IoT, edge and cloud space.

*Contribution to Research Challenge 2:* The ENACT DevOps Framework will facilitate the operations of SIS in a trustworthy and agile fashion. To efficiently maintain and evolve the system according to system requirements, the ENACT DevOps Framework will provide and exploit online learning mechanisms (e.g., extending the ones proposed in [15]). This will allow dynamic self-improvement of the adaptation mechanisms of the SIS, thereby taking into account itselfcontext changes and behavioural drift. To ensure the trustworthy operation of smart applications and in particular the proper actuation on the physical environment, the ENACT DevOps Framework will provide enablers to analyse which actions are permissible in the given run-time context, thereby avoiding potential conflicting actuations and adaptations. Moreover, the ENACT DevOps Framework will enable root cause analysis and prediction in smart IoT systems with respect to trustworthiness, QoS requirements and SLAs.

*Contribution to Research Challenge 3:* To strengthen trustworthiness, the ENACT DevOps Framework will provide a set of coherent enablers to: *(i)* continuously validate and test the proper system design and operation; *(ii)* control and adapt the system structure, behavior and infrastructure including operation-time monitoring of the system and its underlying infrastructure (*i.e.*, nodes, devices, virtual machines, software stack, location and ownership of nodes, etc.); *(iii)* enhance robustness and resilience of smart IoT applications; *(iv)* provide new context-aware security and privacy mechanisms for end-to-end security and privacy across IoT, fog, and cloud infrastructures, integrating not only smart preventive security mechanisms but also reactive security measures; and *(v)* a novel risk-driven management support system that allow identifying threats and supports humans in the selection of device and service features that are important to adhere to specific security, location and QoS requirements. Because smart applications may involve actuators that can have direct impact on the physical world, special attention will be given to ensure the consistency of the different actuation behaviours and the management of intelligent behaviour also at the edge and IoT end.

*Contribution to Research Challenge 4:* The ENACT DevOps Framework will provide the missing links to implement a DevOps software process for IoT systems. The core of the ENACT enabler will be kept independent from the underlying implementation choices (*i.e.*, programming languages, libraries, IoT platform, protocols, devices, etc.). A documented plug-in mechanism will allow to specialize the ENACT enablers to exploit the underlying platform for the implementation of continuous integration, deployment, dynamic adaptation, testing, usage analytics, etc. In the ENACT Framework, plugins for various IoT platforms such as SOFIA, FIWARE and TelluCloud will be developed. Targeting SOFIA and FIWARE ensures the broad applicability of ENACT. TelluCloud allow for the validation of the plug-in mechanism on a proprietary platform. The ENACT approach and enablers have to take advantage of existing components as well as allow choosing the best suited IoT platforms for the task. That includes both platforms available today and future ones.

### 3.1 DevOps life-cycle of SIS

The ENACT DevOps approach is to evolve DevOps methods and techniques to support the development and operation of smart IoT systems, which *(i)* involve sensors and actuators and *(ii)* need to be trustworthy.

DevOps practices aim to ensure a rapid and efficient value delivery to market. DevOps ideas promote a tight collaboration between the developers (Dev) and the teams that deploy and operate the software systems (Ops). DevOps seeks to decrease the gap between a product design and its operation by introducing software design and development practices and approaches to the operation domain and vice versa. In the core of DevOps there is automation and continuous processes supported by different tools at various stages of the product life-cycle. In particular, the ENACT DevOps Framework will support the DevOps practices during the development and operation of trustworthy smart IoT systems and provide innovations and enablers that will feature trustworthy IoT systems related to seven stages of the process as depicted in Figure 1.

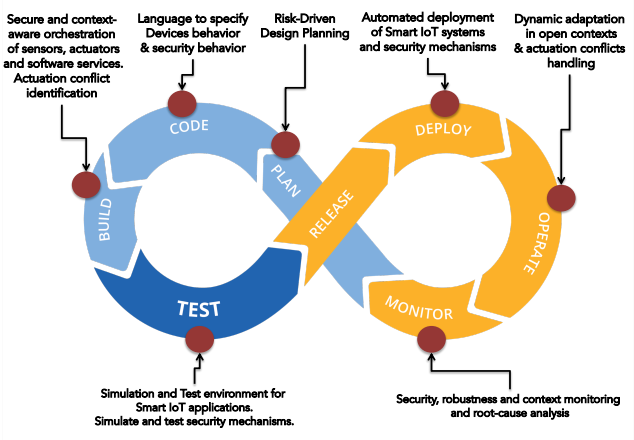


**Figure 1.** ENACT support of DevOps for trustworthy smart IoT systems

**Plan:** The ENACT approach is to support the planning of IoT systems development cycles as well as the smooth transition towards the code stage, introducing a new enabler for risk-driven and context-aware selection of the most relevant and trustworthy devices and services to be used in the future stages. *(Research Challenge 1)*

**Code:** The ENACT approach is to leverage the model-driven engineering approach and in particular evolve recent advances of the ThingML [6] language and generators to support modelling of system behaviours and automatic derivation across vastly heterogeneous and distributed devices at the IoT and edge end. *(Research Challenge 1)*

**Build and Deploy:** The ENACT approach is to provide a new deployment modelling language to specify trustworthy and secure orchestrations of sensors, actuators and software components, along with the mechanisms to identify and handle potential actuation conflicts at the model level. The deployment engine will automatically collect the required software components and integrate the evolution of the system into the run-time environment across the whole IoT, edge and cloud space. *(Research Challenge 1 & 2)*

**Test:** Targeting the constraints related to the distribution and infrastructure of IoT systems, ENACT enablers will allow continuous testing of smart IoT systems in an environment capable of emulating and simulating IoT and edge infrastructures. This system will also be able to simulate some basic attacks or security threats. *(Research Challenges 1 & 3)*

**Operate:** The ENACT approach is to provide enablers for the automatic adaptation of IoT systems based on their run-time context, reinforced by online learning. Such automatic adaptation will address 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. *(Research Challenge 2)*

**Monitor:** The ENACT approach is to deliver innovative mechanisms to observe the status and behaviour of the running IoT systems for quality assurance and root cause analysis, and support the testing of these systems at run-time. *(Research Challenge 2)*

In addition to the DevOps related contributions identified above, the ENACT DevOps Framework will provide specific cross-cutting innovations related to trustworthiness, which can be seamlessly applied, in particular based on the following ENACT concepts:

**Resilience and robustness:** The ENACT approach is to provide novel solutions to make the smart IoT systems resilient by providing enablers for diversifying IoT service implementations, and deployment topologies (*e.g.*, implying that instance of a service can have a different implementation and operate differently, still ensuring consistent and predictable global behavior). This will lower the risk for privacy and security breaches and significantly reduced impact in case of cyber-attack infringes. *(Research Challenge 3)*

**Security, privacy and identity management:** The ENACT approach is to provide support to ensure end-to-end security of trustworthy SIS. This does not only include smart preventive security mechanisms but also the continuous monitoring of *(i)* security metrics and *(ii)* the context with the objective to trigger reactive security measures. *(Research Challenge 3)*

In the following section we present the design of the ENACT DevOps Framework that will support the DevOps life-cycle of trustworthy smart IoT systems.

## 4 The ENACT DevOps Framework

The current initial ENACT DevOps Framework is designed as depicted in Figure 2 and consists of some experimental prototypes of the enablers. This section describes the design of the ENACT DevOps Framework.

The ENACT DevOps Framework is designed as a set of loosely coupled enablers that can be easily integrated with existing IoT platforms via a plug in mechanism. The ENACT enablers are categorized in three groups as follows: *(i)* the toolkit for the continuous delivery of smart IoT systems, *(ii)* the toolkit for the agile operation of smart IoT systems, and *(iii)* the ENACT facilities for trustworthiness. The set of enablers can be seamlessly combined and they can easily integrate with existing IoT platform services and enablers.

### 4.1 ENACT Continuous Delivery toolkit

The designed ENACT DevOps Framework includes two enablers that improve the continuous delivery of smart IoT systems, with a specific focus on *(i)* agile and continuous evolution and *(ii)* early-detection of issues in the software development process. A particular concern is to support the testing of smart IoT systems and the gradual migration from the test to the operation environment.

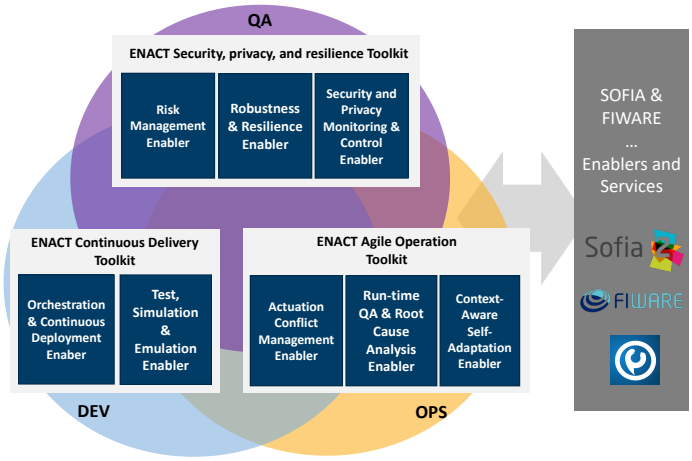


**Figure 2.** The ENACT Framework architecture

*Orchestration and Continuous Deployment Enabler:* This ENACT enabler aims to facilitate the engineering and continuous deployment of smart IoT systems, allowing decentralized processing across the heterogeneous IoT, edge and cloud space. Specifically, a domain-specific modelling language facilitates the design of smart IoT systems by supporting the modelling derivation and orchestration of trustworthy IoT, edge and cloud services that are enabled for context-aware and risk-driven assimilation of the most proper mechanisms for trustworthy execution. The language is designed to come with an execution engine that will support the automatic deployment of software components over IoT, edge and cloud resources, building on work providing a multicloud deployment engine [7]. The enabler is designed to leverage recent techniques for ensuring the proper isolation of each individual service (*i.e.*, containerization using technologies such as Docker[24], rkt[25], or Triton[26]). In addition, it embeds mechanisms to identify actuation conflicts (*i.e.*, conflicts related to concurrent access to a same actuator or due to conflicting modifications of the environment - *e.g.*, opening the windows may conflict with the action of turning off the heating system in order to regulate the temperature) and to generate and deploy the respective controller (*i.e.*, software components whose role is to manage access to the actuators).

*Test, Emulation and Simulation Enabler:* IoT systems need to cope with the uncertainty related to the physical world (*e.g.*, communication links may fail, nodes may run out of

[24] https://www.docker.com
[25] https://coreos.com/rkt
[26] https://docs.joyent.com/public-cloud

battery, etc.). The delivery model advocated to manage this uncertainty should provide proper support to assess the system's behaviour and trustworthiness early in the life cycle. ENACT will deliver an enabler to test smart IoT systems that includes simulation and emulation of IoT services and devices. In particular, a hardware abstraction layer (HAL) will be in charge of offering physical resources and sensors as software services. In addition, this enabler may be based on self-learning IoT virtualizers to automatically model IoT elements and facilitate testing. Finally, this system will also be able to simulate some basic attacks or security threats

### 4.2 ENACT Agile Operation toolkit

We have designed three innovative enablers to significantly reduce the burden of managing and maintaining smart IoT systems. Key requirements are *(i)* to ensure the trustworthiness of such systems under operation and *(ii)* to automate operation activities as much as possible.

*Context-Aware Self-Adaptation Enabler:* Because anticipating all possible context situations that smart IoT systems may encounter during its operation is not possible, it is difficult for software developers to determine how a run-time adaptation of the system may impact the satisfaction of the system behaviour and of the interactions with the environment. To address this challenge, this enabler is designed to apply online learning techniques to improve the way a smart IoT system adapts during its operation. Online learning means that learning is performed at run-time, taking into account observations about the actual system execution and system context. Online learning incrementally updates the smart IoT system's knowledge base; *e.g.*, its adaptation rules or the models based on which adaptation decisions are made.

*Root Cause Analysis Enabler:* This enabler offers a resilient root cause analysis system, which is designed to leverage edge infrastructures to perform diagnostics on a segregated part of the system without connectivity to a centralized diagnostic module or service. With the system behaviour models created for this analysis, this enabler is able to predict future states of the system and provide warnings and recommendations.

*Context Monitoring and Actuation Conflict Management Enabler:* Because of the uncertain, dynamic, and partially known nature of the physical environment, it is very difficult or even illusory to assess at run- time the conformity of the effects of actions in this environment with deterministic models. This enabler is designed to provide a set of observers to monitor the behavioural drift of smart IoT systems that may arise when operating in such open context. In addition, it is designed to exploit the computed drift measure to dynamically adjust the behaviour of the system. This enabler relies on the finite state machines composition theory and model checking tools. By means of deterministic models of the devices, their physical context along with some predefined constraints, a set of device controllers (*i.e.*, software components whose role is to manage access to the actuators) will be generated at design-time and deployed to prevent conflicting and antagonist actions that may occur at run-time.

### 4.3 ENACT Trustworthiness Toolkit

The ENACT DevOps Framework will deliver a set of enablers addressing specific cross-cutting trustworthiness concerns such as ensuring proper robustness, security and privacy of smart IoT systems.

*Robustness and Resilience Enabler:* Recent multidisciplinary research on software engineering and ecology suggests that a promising way to approach resilience in software systems is to diversify software, and system topology [27] , implying that each instance of a service has a different implementation and operates differently, still ensuring that its global behaviour is consistent and predictable. Thus, instead of exposing the very same code and topology in millions of instances, each individual instance can be unique, making the overall system more resilient. Currently, the digitalisation is typically based on commonalities and replications, which can have some severe implications. For example, if a system can break into one "keyless" car, it allows to steal any car that uses the same "keyless" system [28]. This ENACT enabler is inspired from nature where diversity has been a key mechanism for resilience for all forms of life, by enabling them to adapt to changes in the environmental conditions and evolve immunity to perturbations. This enabler is designed to automate the introduction and management of diversity in smart IoT systems and builds on recent research developed in particular [29].

*Risk Management Enabler:* This enabler is designed to provide a risk-driven guidance to architects, developers, feasibility study engineers and other potential stakeholders to design the architecture of their IoT systems and thus support them in the selection of devices, IoT services and mechanisms to ensure trustworthy execution of IoT systems. This is achieved by detecting potential vulnerabilities affecting one or more devices and analyzing trade-off between security and trustworthiness level offered by the devices and services, risk and quality impact. This service can be integrated with the ENACT Continuous Delivery Toolkit, to allow its exploitation in an iterative design process for rapid software evolution. The MODAClouds[27] and MUSA[28] Decision Support Systems will serve as baseline and be extended in ENACT with specific focus on *(i)* the IoT domain and *(ii)* trustworthiness mechanisms. We will also study the potential impact that the late detection of these risks may have in the software development process planning.

*Security and Privacy Monitoring and Control Enabler:* This enabler will provide mechanisms to monitor end-to-end the security, privacy of a smart IoT system. A set of relevant metrics will be defined and notifications will be raised when the monitored metrics deviated from the normal (*i.e.*, risk under control) behaviour. This enabler will include mechanisms and tools to support the user data awareness and control in the form of intelligent notification able to provide insights on what is actually the security issue in the IoT environment. We will leverage open source solutions, particularly for network and system levels monitoring, while new innovative solutions will be developed for the application level security and privacy assessment. Finally, this enabler will provide

[27] http://www.modaclouds.eu
[28] http://www.musa-project.eu

mechanisms for controlling the security, privacy and trustworthiness behaviour of smart IoT systems. This includes the early reaction models and mechanisms that address the adaptation and recovery of the IoT application operation in case the monitored metrics deviated from the expected behaviour. A specific focus will be given to the confidentiality and integrity of data and services via end-to-end Context-based Access control and authorization mechanisms for smart IoT systems. This includes the early reaction models and mechanisms that address the adaptation and recovery of the IoT application operation on the basis of the application context. Today, no protocol can deliver dynamic authorization based on context for both IT and OT (operational technologies) domains. This work will advance state-of-the art mechanisms and leverage on Evidian's IoT access control solutions.

## 5 Conclusion

We presented a set of 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. The ENACT DevOps Framework will offer a set of novel solutions to address these challenges. Most of these enablers will be delivered as open source artefacts.

## References


1. IEC: IoT 2020: Smart and secure IoT platform. IEC white paper (2016)
2. NESSI: Cyber physical systems: Opportunities and challenges for software, services, cloud and data. NESSI white paper (2015)
3. NESSI: SOFTWARE CONTINUUM: Recommendations for ICT Work Programme 2018+. NESSI report (2016)
4. Humble, J., Farley, D.: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional (2010)
5. Taivalsaari, A., Mikkonen, T.: A roadmap to the programmable world: software challenges in the iot era. IEEE Software **34**(1) (2017) 72–80
6. 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), IEEE (2016) 107–116
7. Ferry, N., Song, H., Rossini, A., Chauvel, F., Solberg, A.: CloudMF: applying MDE to tame the complexity of managing multi-cloud applications. In: IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC), 2014, IEEE (2014) 269–277
8. Van Ommering, R., Van Der Linden, F., Kramer, J., Magee, J.: The koala component model for consumer electronics software. Computer **33**(3) (2000) 78–85
9. Fassino, J.P., Stefani, J.B., Lawall, J.L., Muller, G.: Think: A software framework for component-based operating system kernels. In: USENIX Annual Technical Conference, General Track. (2002) 73–86
10. Fouquet, F., Morin, B., Fleurey, F., Barais, O., Plouzeau, N., Jezequel, J.M.: A dynamic component model for cyber physical systems. In: Proceedings of the 15th ACM SIGSOFT symposium on Component Based Software Engineering, ACM (2012) 135–144
11. De Lemos, R., Giese, H., Müller, H.A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N.M., Vogel, T., et al.: Software engineering for self-adaptive systems: A second research roadmap. In: Software Engineering for Self-Adaptive Systems II. Springer (2013) 1–32

12. Esfahani, N., Kouroshfar, E., Malek, S.: Taming uncertainty in self-adaptive software. In: Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, ACM (2011) 234–244
13. Fredericks, E.M., DeVries, B., Cheng, B.H.: Autorelax: automatically relaxing a goal model to address uncertainty. Empirical Software Engineering **19**(5) (2014) 1466–1501
14. Esfahani, N., Elkhodary, A., Malek, S.: A learning-based framework for engineering feature-oriented self-adaptive software systems. IEEE transactions on software engineering **39**(11) (2013) 1467–1493
15. Sharifloo, A.M., Metzger, A., Quinton, C., Baresi, L., Pohl, K.: Learning and evolution in dynamic software product lines. In: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, ACM (2016) 158–164
16. Metzger, A., Di Nitto, E.: Addressing highly dynamic changes in service-oriented systems: Towards agile evolution and adaptation. Software Design and Development: Concepts, Methodologies, Tools, and Applications (2013) 164
17. Aggarwal, C.C.: Managing and mining sensor data. Springer Science & Business Media (2013)
18. Dastjerdi, A.V., Gupta, H., Calheiros, R.N., Ghosh, S.K., Buyya, R.: Fog computing: Principles, architectures, and applications. arXiv preprint arXiv:1601.02752 (2016)
19. Niggemann, O., Biswas, G., Kinnebrew, J.S., Khorasgani, H., Volgmann, S., Bunte, A.: Data-driven monitoring of cyber-physical systems leveraging on big data and the internet-of-things for diagnosis and control. In: DX@ Safeprocess. (2015) 185–192
20. Zasadziński, M., Muntés-Mulero, V., Solé, M., Carrera, D.: Fast root cause analysis on distributed systems by composing precompiled bayesian networks. Proc. World Congr. on Engineering and Computer Science **1** (2016) 464–469
21. OWASP: Internet of Things Top Ten. Technical report (2014)
22. Kolias, C., Stavrou, A., Voas, J., Bojanova, I., Kuhn, R.: Learning internet-of-things security" hands-on". IEEE Security & Privacy **14**(1) (2016) 37–46
23. Cvitić, I., Vujić, M., Husnjak, S.: Classification of security risks in the iot environment. In: 26th International DAAAM Symposium on Intelligent Manufacturing and Automation. (2016) 731–740
24. Kecskemeti, G., Casale, G., Jha, D.N., Lyon, J., Ranjan, R.: Modelling and simulation challenges in internet of things. IEEE Cloud Computing **4**(1) (2017) 62–69
25. D'Angelo, G., Ferretti, S., Ghini, V.: Simulation of the internet of things. In: High Performance Computing & Simulation (HPCS), 2016 International Conference on, IEEE (2016) 1–8
26. D'Angelo, G., Ferretti, S., Ghini, V.: Multi-level simulation of internet of things on smart territories. Simulation Modelling Practice and Theory **73** (2017) 3–21
27. Allier, S., Barais, O., Baudry, B., Bourcier, J., Daubert, E., Fleurey, F., Monperrus, M., Song, H., Tricoire, M.: Multitier diversification in web-based software applications. IEEE Software **32**(1) (2015) 83–90
28. Verdult, R., Garcia, F.D., Ege, B.: Dismantling megamos crypto: Wirelessly lockpicking a vehicle immobilizer. In: USENIX Security Symposium. (2013) 703–718
29. Baudry, B., Monperrus, M., Mony, C., Chauvel, F., Fleurey, F., Clarke, S.: Diversify: Ecology-inspired software evolution for diversity emergence. In: Proc. of the Int. Conf. on Software Maintenance and Reengineering (CSMR), Belgium (2014) 444–447