TOWARDS ADAPTIVE MONITORING

for

SELF-ADAPTIVE SYSTEMS

EDITH ZAVALA

TOWARDS ADAPTIVE MONITORING *for* SELF-ADAPTIVE SYSTEMS

EDITH ZAVALA

Thesis supervised by

DR. XAVIER FRANCH[1] AND DR. JORDI MARCO[2]

[1]Department of Service and Information System Engineering

[2]Department of Computer Science

PhD in Computing program

Universitat Politècnica de Catalunya (UPC) – Barcelona Tech

Abstract

Nowadays, most of the approaches supporting self-adaptive systems rely on static feedback control loops for managing their adaptation process. One of the most popular feedback loops is the MAPE-K loop. In this loop, the Monitor element plays a crucial role since the quality of the monitoring data (e.g., timeliness, freshness, accuracy, availability, etc.) affects directly the performance of the rest of the elements of the loop, and in consequence the quality of the resulting adaptation decisions. Assuming static feedback loops implies that the structure and behavior of the elements of the loop should be determined at design time and cannot change at runtime, i.e., in the case of the Monitor, systems' owners should know everything to be monitored at design time. If that is the case, current self-adaptive systems would not be able to react to unpredictable runtime events such as faults or changing requirements. Motivated by this fact, in this thesis we present HAFLoop (Highly Adaptive Feedback control Loop), an architectural proposal that extends the MAPE-K feedback loop for enabling and supporting the adaptation of the elements of the loop. We propose a generic structure for the adaptive elements as well as the mechanisms required for coordinating their operation with their adaptation process. In HAFLoop, the generic functionality of the elements of the loop is separated from the custom one, thus the proposed architecture can be reused by any approach for designing and developing self-adaptive systems with adaptive feedback loops. Given its importance, for validating HAFLoop, we focus on the adaptation of the Monitor element of the loop. Experiments are executed in the domain of smart vehicles and run in both simulation and real environments.

Acknowledgments

This work would not have been possible without the support of many people. I would like to express my sincere gratitude to Dr. Xavier Franch and Dr. Jordi Marco for guiding me in the realization of this thesis, for their invaluable support, continuous motivation, and patience. I would also like to thank my friends and colleagues, who helped me during all these years and contributed to improve this work. Finally, I would like to thank my family for all the love and unconditional support they give me in every project that I decide to undertake. Especially, I would like to thank Mario for all the stimulating discussions, technical and spiritual support.

This work has been possible thanks to the funding of the National Council for Science and Technology (CONACYT), Spanish projects EOSSAC (TIN2013-44641-P) and GENESIS (TIN2016-79269-R); the SUPERSEDE project, funded by the European Union´s Information and Communication Technologies Programme (H2020) under grant agreement no. 644018; and the SALI project, funded by the Swedish openresearch@astazero program (call 4 - 20180430).

Table of contents

1

Introduction

1.1 Context and terminology

The complexity of modern software systems has increased dramatically over the years and is continuing to do so. Users can access software applications using a variety of devices and since mobile technologies are on the rise, applications are becoming ubiquitous in our society. In order to deal with the great diversity of execution contexts (i.e., user profiles, system faults, changing environmental conditions and user needs, etc.), modern software systems monitor themselves and their environment, and respond to runtime changes through adaptation. In this process, the monitoring task plays a crucial role since the quality of the monitoring data, i.e., its timeliness, freshness, accuracy, availability, etc., affects directly the adaptation decisions. The research of this thesis explores how this quality can be ensured at runtime. In order to do that, two main research fields of Software Engineering have been studied: Self-Adaptive Systems and Adaptive Monitoring. In the rest of this section, these two fields are explained in order to provide a general context and introduce the terminology used along this document.

► Self-adaptive systems

Modern software systems such as smart cities, smart vehicles, and mobile apps are extremely capable, mimicking natural systems' characteristics such as intelligence, rationality, ability to learn, anticipation, and automatic adaptation. This kind of systems are called *Self-Adaptive Systems* (SASs) [1], [2]. Concretely, as defined by Cheng et al.:

> *"A SAS is a system able to automatically modify itself in order to respond to changes in its environment and the system itself"*
>
> Cheng et al., *Software Engineering for Self-Adaptive Systems: A Research Roadmap* [3]

SASs are composed of two main parts: an autonomic manager (AM), also known as the adaptation logic, and the managed elements (MEs), also known as managed resources [2], [4]. The MEs are the components of the software system that provide the main functionality and can be adapted at runtime. While, the AM corresponds to the control unit that manages the MEs' adaptation process [2]. In practice, AMs are implemented through feedback control loops. One of the most popular feedback loops is the MAPE-K loop. The MAPE-K loop has been firstly introduced by Kephart and Chess [5] with their notion of an autonomic element which culminated in IBM's architectural blueprint for autonomic computing and the Autonomic Computing Reference Architecture (ACRA)[6]. The MAPE-K loop consists of five elements that originate its name: Monitor, Analyzer, Planner, Executer, and a Knowledge base (see Figure 1). Below, we describe the function of each MAPE-K element, based on IBM's proposal [6]:

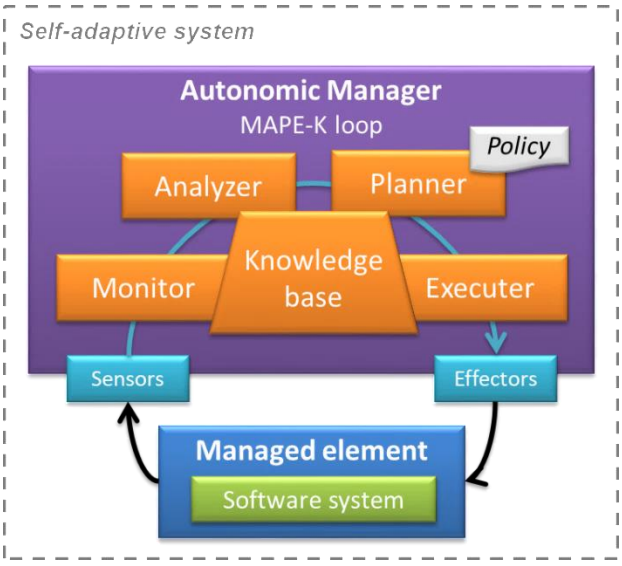


Figure 1: MAPE-K feedback control loop

- ***Monitor***. Its function is to provide the mechanisms that collect, aggregate, filter and report details (such as metrics and topologies) collected from MEs, users, the environment, etc., using a Sensors interface.
- ***Analyzer***. Based on the contextual data gathered by the Monitor element, the Analyzer provides the mechanisms that correlate and model complex situations (for example, time-series forecasting and queuing models). These mechanisms allow the feedback loop to learn about the environment and help predict future situations.
- ***Planner***. Using the analysis results, the main function of the Planner element consists in providing the mechanisms that construct the adaptation actions needed to achieve MEs'

goals and objectives. The Planner uses policy information to guide its operation. Adaptation plans are sent to the Executer for being enacted over the MEs.

- ***Executer***. It is in charge of providing the mechanisms that control the execution of an adaptation plan with considerations for dynamic updates using an Effectors interface.
- ***Knowledge base***. It manages different knowledge sources that can contain different types of knowledge (such as data resulting from MAPE elements' operation and management data) for supporting the operation of the whole loop.

Due to the great diversity of applications, intensive efforts from different research fields have been spent on the realization of SASs. However, some challenges regarding the capabilities and engineering of SASs remain open, what makes this domain still an important subject of research [7].

► Adaptive monitoring

Nowadays, the monitoring activity is integrated into software systems' control processes for gathering relevant data at runtime, as it is the case of SASs. Over the years, different methods and techniques for monitoring software systems in a variety of domains have been proposed. Monitoring allows systems' stakeholders checking how their systems progress or behave under different conditions, and reporting on relevant changes. However, it is often expensive and intrusive. Thus, the design of a monitoring system (i.e., the software system that implements monitoring capabilities) usually involves tradeoffs between the impact caused by the action of monitoring and its expected quality of results, such as data accuracy, freshness, coverage, etc.[8], [9]. In addition, a monitoring system is exposed to a diversity of runtime events, e.g., structural or operational changes on the System under Monitoring (SuM), faults on the monitoring system's components or the emergence of new monitoring requirements.

In order to deal with all these challenging factors, software engineers have proposed different approaches for making current monitoring systems adaptive. Proposals have emerged from a variety of research fields (e.g., sensor networks, instrumentation, requirements monitoring). However, although these diverse proposals share most high-level challenges, solutions have been developed, evolved, and kept isolated in those different fields. This hinders the discovery of synergies among the different proposals to adaptive monitoring as well as the standardization of the main field concepts, starting with the adaptive monitoring term itself, and the normalization of the challenges faced.

The adaptation of monitoring systems requires to manage and control their monitoring activity itself [10]. That is, monitoring systems' components and their operation should be supervised somehow as well, in order to determine monitoring systems' state and the adequacy of their data gathering strategies. According to Moui and Desprats [10], a monitoring strategy can be constructed by answering the questions: why do we monitor?, how do we monitor?, what do we monitor?, and when do we monitor?. In Chapter II, we study how the state-of-the-art

approaches support the adaptation of the monitoring systems' strategies as well as their composition.

► Adaptive monitoring for self-adaptive systems

Most of the current approaches supporting SASs focus on the adaptation of the MEs and consider the AM a static component, i.e., the MAPE-K elements of the loop operating in the AM are not able to change their structure or behavior at runtime. If a change has to be done, it must be executed manually offline. This implies to restart the system that executes the AM and probably a long time of unavailability. The adaptation of the AM might be beneficial and even necessary to respond to different runtime situations. For instance, changes in system resources, its environment or its adaptation goals and requirements, as well as to deal with unanticipated events such as faults [11]–[15]. In order to support the adaptation of the AM, SASs must implement correctly a self-* property called self-improvement. As recently defined by Krupitzer et al. [16]:

> *"Self-improvement [...] is the adjustment of the adaptation logic to handle former unknown circumstances or changes in the environment or the managed resources."*
>
> Krupitzer et al., *Comparison of approaches for self-improvement in self-adaptive systems* [16]

From the state of the art analyzed in this research, only a few studies focusing on supporting the self-improvement property of SASs have been identified. Moreover, most of the approaches propose partial and use case-specific solutions. Apart from that, as pointed out by Krupitzer et al. [16], given the novelty of the topic, different research challenges are still open, ranging from adaptation capabilities to software engineering solutions generalizability.

Among the MAPE-K elements composing the AM, the Monitor element plays a crucial role since the quality of the monitored data affects directly the performance of the rest of the elements and in consequence the resulting adaptation of the MEs. Low quality data will produce low quality adaptations. Current approaches considering static AM, are assuming that software systems' stakeholders know everything to be monitored at design time. However, in order to ensure the adequacy and correctness of the Monitor element operation at runtime, adaptation capabilities should be supported, e.g. for adding new measures to collect, updating the sampling rate, or changing the data collection protocol.

1.2 Objectives and research questions

The complexity of handling the adaptation of SASs' AM, concretely the Monitor element, in coordination with the MEs' adaptation process turns out to offer new opportunities to software engineers. Several approaches for supporting adaptive monitoring have been proposed; however, very few of them support adaptive monitoring in SASs. Motivated by this fact, this

thesis explores and experiments with new and existing methods and techniques in order to provide a solution for supporting adaptive monitoring in SASs. Concretely, the main goal of this thesis is:

> *To support the automatic runtime adaptation of self-adaptive systems' autonomic manager, particularly the Monitor element, in order to respond to changes in the managed elements, the environment, and the autonomic manager itself.*

In order to reach our research goal, we first have to understand the needs of current monitoring systems and the current state-of-the-art of the adaptive monitoring topic. The adaptation of the Monitor element of the AM is a specific application of the self-improvement property of SASs. Therefore, understanding how the Monitor and the other elements of the AM are adapted by existing SASs solutions (if any) so far, is also very important for the purposes of this thesis. Understanding the current open research challenges and opportunities will allow us to develop a suitable solution for supporting adaptive monitoring in modern SASs. In order to achieve this, we have defined three main research questions that will be addressed in this thesis (see Table 1).

Table 1
Research questions of this thesis

Id	Research questions
RQ1	How is monitoring systems' adaptation, in general, supported by existing approaches?
RQ2	How is self-adaptive systems' self-improvement supported by existing approaches?
RQ3	How the self-improvement property, particularly autonomic manager's Monitor element adaptation, can be (better) supported in self-adaptive systems?

Below, the main objective of each research question is described:

- RQ1. This RQ aims at investigating on the approaches of existing proposals supporting the adaptation of monitoring systems, and provide a framework of common understanding for the definition of **how to adapt**, monitoring systems.
- RQ2. The objective of RQ2 is to investigate on the approaches of existing proposals supporting the adaptation of the AM in SASs, and provide a framework of common understanding for the definition of **how to improve**, SASs through the implementation of the self-improvement property.
- RQ3. Finally, this RQ aims at investigating on the aspects that can be done (and improved in existing proposal) for understanding **how to support** SASs' self-improvement (particularly, understanding how the adaptation of the AM's Monitor element can be coordinated with loop's normal operation), and develop an architectural solution that correctly supports the process.

1.3 Methodological approach

In order to address the RQs of this thesis, we have adopted a methodological approach based on the approach proposed by Shaw [17] for characterizing software engineering research. The characterization proposed by Shaw is described in terms of research settings, research products, and validation techniques. Below, we describe how the research of this thesis is characterized in each of these terms.

▶ Research settings

Research settings refer to different classes of research problems, i.e., a problem is transferred into a research setting with the aim of finding solutions to it. Shaw lists five research settings corresponding to different types of RQs (see Table 2) [17].

Table 2
Research settings [17]

Research Setting	Sample questions
Feasibility	Is there an X, and what is it? Is it possible to accomplish X at all?
Characterization	What are the important characteristics of X? What is X like? What, exactly, do we mean by X? What are the varieties of X, and how are they related?
Method/Means	How can we accomplish X? What is a better way to accomplish X? How can I automate doing X?
Generalization	Is X always true of Y? Given X, what will Y be?
Selection	How do I decide between X and Y?

The settings of this thesis are characterization, and method/means, i.e.:

- In RQ1 and RQ2, we address the characterization of different approaches for supporting the adaptation of monitoring systems and SASs' AM.
- In RQ3, we define ways and improvements to the existing ways for (better) accomplishing the adaptation of the AM's Monitor element in SASs.

▶ Research products

The research products are the tangible results of the research. Five research products are described by Shaw as well as the ways of how to achieve them (see Table 3) [17]. The research products of this thesis include:

- A qualitative or descriptive model, reporting the state of the art of the adaptive monitoring and SASs' self-improvement topics (RQ1 and RQ2)
- Techniques, on how to correctly support the adaptation of the AM's Monitor element in SASs (RQ3)
- A system, implementing the techniques for the approach validation (RQ3)

Table 3
Research products [17]

Research Product	Research approach or method
Qualitative or descriptive model	Organize & report interesting observations about the world. Create & defend generalizations from real examples. Structure a problem area; formulate the right questions. Do a careful analysis of a system or its development
Technique	Invent new ways to do some tasks, including procedures and implementation techniques. Develop a technique to choose among alternatives.
System	Embody result in a system, using the system development as both source of insight and carrier of results.
Empirical predictive model	Develop predictive models from observed data.
Analytic model	Develop structural (quantitative or symbolic) models that permit formal analysis.

► Research validation

The validation techniques allow evaluate the research results to demonstrate that they satisfy the research settings. In this regard, Shaw proposes a list of five validation techniques (see Table 4) [17].

Table 4
Validation techniques [17]

Technique	Character of validation
Persuasion	A technique, design or example.
Implementation	Of a system or technique.
Evaluation	With respect to a descriptive model, a qualitative model, an empirical quantitative model.
Analysis	Of an analytic formal model, an empirical predictive model.
Experience	Expressed in a qualitative or descriptive model, as decision criteria or an empirical predictive model.

From this list, the validation techniques used in this thesis are:

- *Persuasion*, using examples that illustrate our ideas and proposal
- *Implementation*, of a system to demonstrate the feasibility and adequacy of our solution for supporting adaptive monitoring in SASs
- *Evaluation*, of the system implemented and the proposed approach with a series of use cases

For conducting the research of this thesis, we have designed an iterative process consisting of literature reviews, opportunities and challenges identification, proposal development, system implementation, and approach validation (see Figure 2).

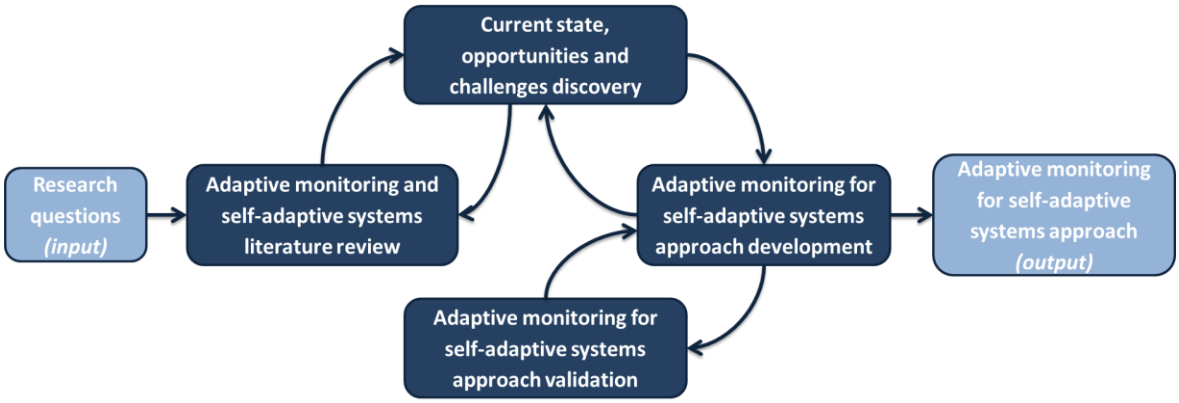


Figure 2: Research process

1.4 Contributions of this thesis

► Contributions of RQ1

The main contribution of this RQ is given by a systematic mapping on adaptive monitoring which attempts to provide a reference for future researches and practitioners in the field, especially to offer a starting point avoiding the development of new proposal that do not align with the current needs and research. Specifically, the study evaluates the current state of the art of adaptive monitoring investigating the following aspects: 1) which are the current proposals and how are they related; 2) how do they define adaptive monitoring; 3) how do they conduct the adaptation of monitoring systems; and, 4) what are the relations among the proposals' solutions. From the research perspective, the following contributions can be remarked:

- Generic definition (missing in the current state-of-the-art) for the term *adaptive monitoring*
- Distribution of the studies about adaptive monitoring based on their demographic characteristics (time, location, etc.)
- Relation among studies about adaptive monitoring from the solution perspective and distribution of them in approaches
- Current state and patterns on approaches for supporting the adaptation of monitoring systems
- Comprehensive overview of the domains where adaptive monitoring has been applied

The contribution of this cross-domain study is to make available a solid and comprehensive baseline for researchers and practitioners in the adaptive monitoring field. Especially, it may help in identifying opportunities of research; for instance, the need of proposing generic and flexible software engineering solutions for supporting adaptive monitoring in a variety of systems. The systematic mapping study has been published at the SCI-indexed journal *Information and Software Technology* (I.F.2017: 2.627) [18].

► Contributions of RQ2

The main contribution of this RQ is given by a series of literature reviews on this topic, which attempt to provide an overview of the current research and requirements of SASs and the AM's adaptation. Specifically, the reviews evaluate the current state of the art investigating the following aspects: 1) which are the current proposals; 2) what components of the AM are adapted; and, 3) how do they conduct the adaptation process. From the research perspective, the contributions of the literature reviews include:

- The identification of methods and techniques utilized so far for supporting SASs' self-improvement
- An analysis of modern SASs' requirements
- The assessment of the applicability of existing proposals in modern SASs

The literature reviews have been conducted at different points of time, aligned with the iterative process we have designed for conducting this thesis research (see Figure 2 in Section 1.3). The contributions of this RQ were published in: two deliverables of the SUPERSEDE H2020 European project [19], [20] and the SCI-indexed journal E*xpert Systems with Applications* (I.F.2017: 3.768)[7].

► Contributions of RQ3

This RQ has different contributions. First, based on the contributions of RQ2, open research challenges in SASs in general and for supporting their self-improvement in particular have been identified. Second, an architectural approach for addressing the still open research challenges affecting SASs' self-improvement and in this way correctly support the adaptation of the AM, especially the adaptation of its Monitor element. Third, a generic and modular framework for supporting adaptive AM in SASs, named HAFLoop, implementing the ideas of our approach. HAFLoop has been validated by different use cases in the domain of smart vehicles. Experiments have been run in both simulation and real environments.

Aligned with our research methodology, several refinement and evaluation iterations have been performed during the development process of our final proposal, which is reflected in the different publications resulting from the contributions of this RQ. Concretely, the first contribution related to this thesis proposal was developed by the applicant in her Master thesis [21] followed by a publication at the demo tool session of the *23rd IEEE International Requirements Engineering Conference* (CORE2018: A) [22]. Later, during the development of this thesis, the contributions of this RQ were published in:

- three deliverables of the SUPERSEDE H2020 European project [23]–[25],
- a report of the SALI Swedish project (openresearch@astazero program) [26],
- the tutorials and poster abstracts session of the *BSR winter school - Big Software on the Run: Where Software meets Data* [27],

- the PhD symposium of the *International Conference on Service-Oriented Computing* (CORE2018: A) [28],
- the SCI-indexed journal *Expert Systems with Applications* (I.F.2017: 3.768) [7]

List of contributions

Table 5 summarizes the list of publications related to this thesis, authored, and co-authored by the applicant.

Table 5
List of publications

Ref.	Authors	Type/Venue	Title	Year	Impact
Published peer-reviewed					
Journal					
[18]	Edith Zavala, Xavier Franch, Jordi Marco	Information and Software Technology	Adaptive Monitoring: A Systematic Mapping	2019	I.F.: 2.627
[7]	Edith Zavala, Xavier Franch, Jordi Marco, Alessia Knauss, Daniela Damian	Expert Systems with Applications	SACRE: Supporting contextual requirements' adaptation in modern self-adaptive systems in the presence of uncertainty at runtime	2018	I.F.: 3.768
Conference					
[28]	Edith Zavala	International Conference on Service-Oriented Computing	Towards Adaptive Monitoring Services for Self-adaptive Software Systems	2017	CORE A
[22]	Edith Zavala, Xavier Franch, Jordi Marco, Alessia Knauss, Daniela Damian	IEEE International Requirements Engineering Conference	SACRE: A Tool for Dealing with Uncertainty in Contextual Requirements at Runtime	2015	CORE A
Other publications					
[27]	Edith Zavala, Xavier Franch, Jordi Marco	BSR winter school - Big Software on the Run: Where Software meets Data	Decision-Making Support for Software Adaptation at Runtime	2016	-
Published non-peer-reviewed					
Technical report					
[26]	Edith Zavala, Christian Berger, Xavier Franch, Jordi Marco	SALI [Project final report]	Smart self-driving vehicle project: Final report	2018	-
[25]	Jesús Gorroñogoitia, Denisse Muñante, Fitsum Kifetew, Angelo Susi, Edith Zavala, Srdjan Stevanetic	SUPERSEDE [H2020 Project Deliverable]	D3.6: Methods and techniques for runtime DM v1	2016	-

[23]	Marc Oriol, Denisse Muñante, Jesús Gorroñogoitia, Anna Perini, Danilo Valerio, Edith Zavala, Quim Motger	SUPERSEDE [H2020 Project Deliverable]	D4.8: Feedback-gathering and monitoring reconfiguration techniques v2	2016	-
[20]	Anna Perini, Danilo Valerio, Denisse Muñante, Edith Zavala, Marc Oriol, Melanie Stade, Norbert Seyff, Jesús Gorroñogoitia,	SUPERSEDE [H2020 Project Deliverable]	D4.7: Feedback-gathering and monitoring reconfiguration techniques v1	2016	-
[24]	Gorroñogoitia, Jesús, Edith Zavala, Marc Oriol, Quim Motger, Srdjan Stevanetic,	SUPERSEDE [H2020 Project Deliverable]	D4.5: Methods and tools to enact software adaptation and personalization v2	2016	-
[19]	Jesús Gorroñogoitia, Danilo Valerio, Tudor Ionescu, Edith Zavala	SUPERSEDE [H2020 Project Deliverable]	D4.4: Methods and tools to enact software adaptation and personalization v1	2016	-
			Other publications		
[29]	Edith Zavala	PhD proposal	Towards Adaptive Monitoring for Self- * Systems: Research Plan	2017	-
[21]	Edith Zavala	Master thesis	Dealing with Uncertainty in Contextual Requirements at Runtime: A Proof of Concept	2015	-

Co-authorship statement: *In all the aforementioned publications in which the applicant figures as first author, she has been the main contributor to all aspects related to self-adaptive systems' adaptation process and self-adaptive systems' self-improvement, under the supervision of the two PhD thesis advisors. Whereas, the contributions of specific research aspects that are out of scope of this thesis (e.g. feedback gathering) were conducted mainly by the other authors. Moreover, in the rest of co-authored publications the applicant has contributed in the aspects related monitoring reconfiguration, decision-making, and adaptation enactment.*

1.5 Structure of this thesis

This thesis is presented in three parts, which correspond to the three RQs exposed in Section 1.2. The first part "*How to adapt: Study on adaptive monitoring*", refers to RQ1, the second part "*How to improve: Study on SASs' self-improvement*", refers to RQ2, and the third part "How to support: Building a SASs' self-improvement architecture" refers to RQ3. Each part presents the contributions we have described in Section 1.4, for each of the RQs. Finally, we

provide the conclusions and future work of this thesis. Table 6 specifies the overview of each chapter.

Table 6
Overview of thesis' chapters

Chapter	Overview
1	Provides the fundamentals and research settings of this thesis
2	Describes a Systematic Mapping conducted to study the different approaches for supporting the adaptation of monitoring systems. It is divided in five main sections. Section 1 and 2 introduce adaptive monitoring and systematic mapping studies, respectively. Section 3 provides the planning for the review, including the threats to validity. Then, Section 4 reports the results of the review. Finally, Section 5 discusses the results of the review.
3	Presents the state of the art in the field of SASs' self-improvement through different Systematic Literature Reviews. It uncovers modern SASs' needs and the different approaches for satisfying such needs. It is divided in five main sections. Section 1 introduces self-adaptation. Section 2 presents open research challenges affecting SASs. Section 3 presents the protocol and results of the SLR conducted for determining the state of the art regarding the open research challenges identified in Section 2. Section 4 presents open research challenges affecting SASs' self-improvement. Section 5 presents the protocol and results of the SLR conducted for determining the state of the art regarding the open research challenges identified in Section 4.
4	Describes the proposed architectural solution for correctly supporting SASs' AM adaptation, in general, and AM's Monitor element adaptation, in particular. It is divided in seven sections. Section 1 described our vision about SASs' AM adaptation. Section 2 describes the architecture. Section 3 presents a proof-of-concept of our proposal. Section 4 presents the implementation of our solution in the form of a framework. Section 5 provides the details of a use case implementation using our framework. Section 6 presents the evaluation of our solution in different environments. Section 7 provides the performance results of the evaluation.
5	Provides the conclusions and future work of this thesis
6	Presents the references list of this thesis

II

How to adapt

Study on adaptive monitoring

Over the years, methods and techniques for monitoring a variety of systems have been proposed. There are approaches proposed for monitoring communication networks (e.g., Liu et al. [30]), buildings' or persons' health (e.g., Kijewski-Correa et al. [31] and Mshali et al. [32], respectively), software systems (e.g., Toueir et al. [33]), environmental conditions (e.g., Alippi et al. [34]), etc. Monitoring allows systems' stakeholders checking how their systems progress or behave under different conditions, and reporting on relevant changes. However, it is often expensive and intrusive. Thus, the design of a monitoring system (i.e., the software system that implements monitoring capabilities) usually involves tradeoffs between the impact caused by the action of monitoring and its expected quality of results, such as data accuracy, freshness and coverage, among others [8], [9]. In addition, a monitoring system is exposed to a diversity of runtime events, e.g., structural or operational changes on the System under Monitoring (SuM), faults on the monitoring system's elements or the emergence of new monitoring requirements.

In order to deal with all these challenging factors, software engineers have proposed different approaches for making current monitoring systems *adaptive*. Interesting proposals have emerged from a variety of research fields (e.g., sensor networks, instrumentation, requirements monitoring). However, although these diverse proposals share most high-level challenges, solutions have been developed, evolved, and kept isolated in those different fields. This hinders the discovery of synergies and reusable components among the different proposals to support adaptive monitoring as well as the standardization of important concepts, starting with the

adaptive monitoring term itself. This contribution aims at uncovering and characterizing existing approaches supporting the adaptation of monitoring systems.

In order to achieve this goal, we have conducted a systematic mapping study (SMS) for identifying the primary studies on adaptive monitoring published in academic venues. We have retrieved and selected the studies conducting a rigorous protocol, defined later in this chapter, which follows the guidelines presented by Petersen et al. [35] and Kitchenham & Charters [36]. For analyzing the identified studies, we have designed five high-level research questions (RQs) which we have divided into 18 research sub-questions. To extract data from these studies, we have used a qualitative analysis approach based on the method describe by Miles et al. [37]. After the qualitative analysis, we have applied Data Mining over the extracted data for identifying patterns in the approaches. Concretely, we have used the rule-based algorithm JRip [38], [39] implemented by the Data Mining tool Weka [40].

The aim of this first contribution of the thesis is to identify and relate existing proposals of adaptive monitoring, characterize them with respect to some criteria, and uncover patterns on how adaptive monitoring is conducted by approaches so far. The importance of this contribution lies mainly along two lines, namely providing an overview of existing adaptation processes and providing a generic definition for the *adaptive monitoring* term, not found in the studies surveyed. The systematic mapping study has been published at the SCI-indexed journal *Information and Software Technology* (I.F.2017: 2.627) [18].

2.1 Introduction to adaptive monitoring

Adaptive (and self-adaptive) systems have emerged as a response to the increasing complexity of modern software systems. Nowadays, complex software systems are enabled with adaptation capabilities that allow them to respond to changes in the environment and the system itself. Given its wide range of application, this kind of systems has been subject of considerable research effort. For instance, Krupitzer et al. [2] and Salehie et al. [41] have presented extensive surveys on self-adaptive systems in general as well as taxonomies for unifying and improving the understanding of the concepts present in this research area. Given their research objectives, none of these works has analyzed how the adaptation process should be conducted, or may differ in a specific type of adaptive system, such as monitoring systems.

Nowadays, the monitoring activity is integrated into control processes for gathering relevant data that is later analyzed by other software systems or the SuM administrators. The results of the analysis are mainly used for determining the state of the system and deciding whether any action (e.g., administering a medication when monitoring a person's health, or modifying a software service behavior in a nuclear plant) should be taken for keeping the SuM under control. Although some works consider the data gathering and analysis activities as part of a whole monitoring system (e.g., works by Bukenya et al. [42] and Ramirez et al. [9]), in this SMS we

differentiate between them and focus on approaches that specifically support the adaption of the data gathering activity.

2.2 Introduction to systematic mapping studies

Systematic mapping studies or scoping studies are designed to give an overview of a research area through classification and counting contributions in relation to the categories of that classification [36], [43]. It involves searching the literature in order to know what topics have been covered, and where the literature has been published [43]. SMSs share some commonalities with another type of empirical instrument, namely systematic literature reviews (e.g., with respect to searching and study selection). However, according to Petersen et al. [35], they are different in terms of goals and approaches to data analysis. While systematic literature reviews aim at synthesizing evidence, considering its strength, SMSs are primarily concerned with structuring a research area [35].

In order to ensure the quality of systematic reviews, a precise and rigorous methodology for conducting the review process has to be used. For this purpose, we have followed the widely used guidelines proposed by Kitchenham & Charters [36] in conjunction with the updated ones for SMSs proposed by Petersen et al. [35]. The review process consists of three main phases:

- *Planning the review*. During this phase, all the decisions relevant to conducting the study are made. This includes the identification of the need for a review, the definition of the protocol for identifying primary studies and extracting the relevant data, and the definition of the visualization instruments and the validity threats of the study.
- *Conducting the review*. In this phase, the review process as defined during the planning phase has to be implemented. This process is iterative and may require revisions. It is recommended to record the information at all stages of the process.
- *Reporting the mapping*. Finally, this phase consists in reporting the results of the review. It includes specifying the dissemination mechanisms, the format of the report and the evaluation of the process.

2.3 Planning the review

According to Petersen et al. [35] and Kitchenham & Charters [36], the planning phase of the review process consists of five main activities: need for a review identification and scoping, study identification, data extraction and classification, visualization and analysis of validity threats. In this section, we describe how we have performed each of these activities in our SMS. As recommended by Petersen et al. [35], some activities have been further split into sub-activities.

2.3.1. Need identification and scoping

The need identification and scoping activity has been divided in two sub-activities: need for a review identification and research questions definition.

► Need for a review identification

Before carrying out any systematic literature study, researchers should identify and evaluate any existing systematic review on the topic of interest [35]. Hence, in order to identify secondary studies on adaptive monitoring, we have followed a search protocol analogous to the main one presented in the study identification phase of this SMS (see Section 2.3.2). In consequence, we have searched for existing reviews once the protocol was defined and before the SMS was conducted. In short, we have built a search string as a conjunction of population and intervention, as recommended by Kitchenham & Charters [36], and performed an automatic search on the databases of IEEE Xplore, ACM, Scopus and Inspect/Compendex (Engineering Village). We have selected these databases based on the experience reported by Dybå et al. [44] and the results obtained by Petersen et al. [35] using them.

In software engineering, the population may refer to a specific software engineering role, a category of software engineer, an application area, or an industry group [36]. In our context, the population corresponds to studies in the application area of adaptive monitoring (see Table 7). On the other hand, the intervention corresponds to a software methodology/tool/technology/procedure that addresses a specific issue [36]. In our case, the intervention is systematic mappings (see Table 7). In order to increase the number of results, from each main term, we have defined a set of synonyms, variants, and acronyms, which are shown in Table 7. Wildcards have not been used because: 1) some databases do not support the number of wildcards per search we would require; 2) in this way, we dramatically reduce the number of noisy studies. We have constructed the search string by applying the Boolean OR operator to link the Population terms and Intervention terms presented in Table 7 separately, and a Boolean AND operator to link these two resulting substrings.

The search resulted in 271 papers. Then, we have applied a study selection protocol similar to the one applied in our SMS. The only difference is the inclusion/exclusion criteria we have used for selecting the studies of interest. In this case, the inclusion criteria that have been applied were:

- Studies present summaries of adaptive monitoring approaches
- Studies are in the fields of computer science or engineering
- Studies were published until 2016

For excluding studies, we have applied the following criteria:

- Studies present non-peer reviewed material
- Studies not written in English
- Studies not accessible in full-text

Table 7
Search string terms.

Type	Main term	Alternative terms (Synonyms/Variants/Acronyms)	
Population	Adaptive monitoring	adaptive monitor adaptive monitors adaptable monitoring adaptable monitor monitor adaptation monitoring adaptation reconfigurable monitor reconfigurable monitoring monitoring reconfiguration dynamic monitor dynamic monitors dynamic monitoring monitoring evolution monitor evolution monitors evolution	evolving monitoring evolutionary monitoring monitoring customization customized monitor customized monitors customized monitoring customised monitoring monitoring personalization personalized monitors personalized monitoring personalised monitoring reactive monitoring reactive monitors proactive monitoring
Intervention	Systematic mappings	systematic mapping state of the art SLR review	

After applying the selection protocol, we have not found any secondary study on the adaptive monitoring topic, neither in general or in a particular research field. However, when performing the snowballing process in our SMS, we have been able to identify one related work [45]. Although, this work is not focused on adaptation and surveys only approaches supporting energy-efficient wireless sensor networks, we have considered it worth to mention since it has been the only review we have found related to our work. As we will explain later in Section 2.3.2, the approaches cited in this survey that provide energy-conservation through the adaptation of the data gathering activity have been considered in our SMS.

► Research questions definition

Given the lack of secondary studies, conducting a SMS in the adaptive monitoring topic is important and justified. In order to provide a comprehensive overview of the current state of the art, for this SMS, we have designed five high-level RQs divided into 18 research sub-questions (see Table 8).

Table 8
Research questions of the review.

Research Question	Sub-question	
RQ1. What is adaptive monitoring?	RQ1.1	What are the terms related to the term "adaptive monitoring"?
	RQ1.2	Are there specific definitions of adaptive monitoring?
RQ2. What are the demographic characteristics of the studies about adaptive monitoring?	RQ2.1	When are the studies published?
	RQ2.2	Where are the studies published?
	RQ2.3	How are publications distributed between academy and industry?
	RQ2.4	How publications are geographically distributed?
	RQ2.5	How are the studies organized into approaches for adaptive monitoring?
RQ3. What is proposed by adaptive monitoring approaches?	RQ3.1	What type of contributions is presented?
	RQ3.2	How generic are the solutions presented?
RQ4. How adaptive monitoring is conducted by the approaches?	RQ4.1	What is the purpose of adaptation?
	RQ4.2	What is adapted?
	RQ4.3	What triggers adaptation?
	RQ4.4	How analysis is performed?
	RQ4.5	How adaptation decisions are made?
	RQ4.6	How adaptation decisions are enacted in the monitoring system?
	RQ4.7	What type of adaptation is executed?
RQ5. How adaptive monitoring approaches are evaluated?	RQ5.1	What type of evaluation is performed?
	RQ5.2	In which type of systems is the evaluation performed?

2.3.2. Study identification

The study identification activity has been divided into three sub-activities: search string construction, literature sources identification, and study selection.

► Search string construction

The aim of the search process is to find as many primary studies related to the RQs as possible using an unbiased search strategy. In order to build the search string, we have followed again the recommendation of Kitchenham & Charters [36] and created the string as a conjunction of population and intervention. As in the secondary studies' search, our population is composed by studies in the application area of adaptive monitoring. What has changed in this search is the intervention: we are now interested in approaches supporting adaptive monitoring and not in SMSs. In order to increase the number of results, we have defined a set of synonyms and variants for the main search terms (i.e., adaptive monitoring and approaches). In the case of the population, we have reused the alternative terms identified in Section 2.3.1 (see Table 7).

While evaluating the articles resulting from the search of Section 2.3.1, we have noticed that the *dynamic monitor, dynamic monitors* and *dynamic monitoring* terms have been utilized by some of the studies for referring to the continuous runtime monitoring of dynamic factors (e.g.,

in works by Bukenya et al. [42] or Magalhães et al. [46]). Or, as an adjective to describe how the adaptation process is actually conducted (e.g., in works by Clark et al. [47] or Jeswani et al. [48]). Thus, in order to reduce the amount of noisy papers, we have decided to do not consider these terms for the search string of the SMS. Regarding the intervention, we have identified the alternative terms: approach, method, framework, and technique. The search string has been constructed using the terms and the Boolean OR and AND operators as we have done in Section 2.3.1.

► Literature resources identification

In order to identify primary studies, researchers can perform either automatic search through the usage of scientific databases or manual search through gathering the studies from specific known journals and conferences of the target field. Both approaches present advantages and drawbacks. The most common way of searching is the automatic search, followed by the manual search [35]. However, in this SMS, this was not possible since we were not able to identify any relevant dedicated conference or journal in the specific field of adaptive monitoring (neither before nor after conducting the data extraction process). For this reason, similarly to Petersen et al. [35], we have decided to conduct an automatic search and complement it with a backward snowball sampling of all studies selected after full-text reading.

In order to select the databases for conducting the automatic search, we have followed the same criteria as in Section 2.3.1, since we have not found any other secondary study in the adaptive monitoring topic for guiding the search. Thus, the databases used in the SMS are IEEE Xplore, ACM, Scopus, and Inspect/Compendex (Engineering Village). As recommended by Petersen et al. [35], we have used a tool for managing the references extracted from the databases and a tool for recording extracted data. Concretely, we have used the reference management tool Mendeley and the qualitative data analysis tool Atlas.ti® (www.atlasti.com).

► Study selection

In order to select the final set of studies, we have designed a study selection strategy that consists of five stages (see Figure 3). Our strategy is an adaptation of the steps proposed by Petersen et al. [35] and Kitchenham & Charters [36]. In Figure 3, we provide an overview of our strategy and the number of papers resulting in each stage. Figure 3 also details the backward snowballing process we have conducted in the last stage of our study selection strategy.

The exclusion of studies has been done based on titles and abstracts, as well as full-text reading. In order to identify as many primary studies as possible, we have also added studies through backward snowballing. The application of the inclusion and exclusion criteria has been conducted by the applicant. Along the process, periodical meetings have been held with the rest of the authors for discussing and refining the final set of included and excluded papers. The following inclusion criteria have been applied to the studies:

- Studies present a solution (i.e., approach, method, framework, technique or others) for supporting adaptive monitoring.
- Studies are in the fields of computer science or engineering.
- Studies were published until 2016.

Studies fulfilling the following criteria have been excluded:

- Studies are secondary studies.
- Studies present work in progress.
- Studies present non-peer reviewed material.
- Studies are not written in English.
- Studies are not accessible in full-text.
- Studies are books, books reviews, or grey literature.

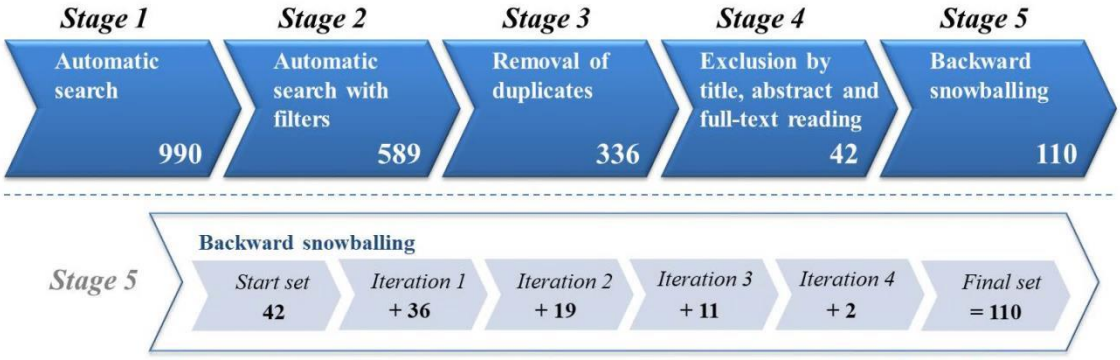


Figure 3: Study selection strategy

Below, we provide the details of each of the stages of the study selection strategy shown in Figure 3.

– *Stage 1 - Automatic search.* This stage corresponds to the automatic search on the digital databases we have detailed before. In this stage, 990 primary studies have been identified. Table 9 shows how many studies have been extracted per database (see column *Search results*).
– *Stage 2 - Automatic search with filters.* After performing the automatic search, we have applied a set of filters that some of the digital libraries offer for automatically excluding studies that are not of our interest. The filters correspond to some of the inclusion/exclusion criteria we have listed before in this section. Table 9 shows the filters we have used in each database and the resulting number of articles after applying those filters (see column *Filtered search results*). In this stage, 401 papers have been automatically discarded, resulting in 589 primary studies.
– *Stage 3 - Removal of duplicates.* From the 589 papers identified in the previous stage, we have automatically removed duplicated studies by using the reference manager Mendeley.

In addition, the applicant has manually reviewed the list of articles in order to identify duplicated records (no detected by Mendeley). As a result, 253 articles have been excluded. That is, after this stage we have ended up with 336 remaining primary studies.

– ***Stage 4 - Exclusion by title, abstract and full-text reading.*** In this stage, the applicant has reviewed all the titles and abstracts and applied the inclusion and exclusion criteria for each study. A paper has been taken to full-text reading when in doubt and discussed with the thesis supervisors. The final set of included and excluded papers has been revised through a series of periodic meetings. After this stage, 294 out of the 336 studies resulting from the previous stage have been excluded, resulting in 42 remaining articles.

Table 9
Number of studies per database with filters applied.

Database	Filters	Search results	Filtered search results
IEEE	No filters applied	95	95
ACM	*Exclude*: 2017	85	84
Scopus	*Exclude*: 2017 *Limit to:* – *Subject Area*: Computer Science, Engineering – *Document Type*: Conference paper, Article – *Language*: English	440	238
Inspect/ Compendex	*Exclude:* 2017 *Limit to:* – *Classification code*: Computer Software, Data Handling and Applications, Computer Applications, Control Systems, Digital Computers and Systems, Computer Systems and Equipment, Automatic Control Principles and Applications, Distributed Systems Software, Software Engineering techniques – *Document type*: Conference article, Journal article, Conference proceeding – *Language*: English	370	172

– ***Stage 5 - Backward snowballing.*** In order to identify as many primary studies as possible, we have conducted a backward snowballing process organized into four iterations (see Figure 3). The process' start set has been composed of the articles that have resulted from Stage 4 (42 articles). While iterating, relevant works have been identified from the reference list of the articles. During the first iteration of the snowballing process, we have identified a secondary study relevant to our SMS [45]. As we have explained in Section 2.3.1, this secondary study surveys approaches supporting energy-efficient wireless sensor networks. Due to the inclusion and exclusion criteria, this study has not been included in our final set of articles. However, since we have identified that some of the surveyed approaches'

solutions involve the adaptation of the data gathering activity, we have considered this secondary study when performing the backward snowballing process. That is, for the second iteration we have included the relevant works identified in the reference list of this secondary study.

Referenced works have been included based on the inclusion and exclusion criteria we have previously defined in this section. Moreover, we have decided to exclude papers published before 2000 (publication year of the oldest start set paper is 2001). Figure 3 shows the number of papers that we have extracted during the process and that fulfill the inclusion criteria (the secondary study mentioned before, identified in iteration 1, has been omitted in the image for the sake of simplicity). As recommended by Wohlin [49], we have finished the snowballing when no new papers fulfilling our criteria have been found. From this stage, 68 papers have been added to the start set, resulting in a final set of 110 relevant primary studies to analyze in our SMS.

2.3.3. Data extraction and classification

In order to extract the data from the primary studies, we have used a qualitative data analysis approach based on the method described by Miles et al. [37]. The qualitative data analysis tool Atlas.ti® has been used for supporting this process and ensuring consistent and accurate extraction of the key information related to the RQs. The extraction process has been performed by the applicant, and reviewed and confirmed by the supervisors. Extracted data has been discussed in a series of periodic meetings scheduled for this purpose. To extract data from the primary studies, we have developed the template shown in Table 10. The qualitative analysis has consisted of the following three steps:

- *Data extraction preparation.* In this step, the 110 primary studies included in our SMS have been imported into a new Atlas.ti® project.
- *First cycle coding.* Codes are labels that assign symbolic meaning to the descriptive or inferential information compiled during a study. They are primarily, but not exclusively, used to retrieve and categorize similar data chunks so the researcher can quickly find, pull out, and cluster the segments relating to a particular RQ, hypothesis, construct, or theme [37]. In order to create the codes of our SMS, we have performed both deductive and inductive coding. First, we have defined a start list of codes from the RQs, i.e., deductive coding. Then, we have added codes that progressively emerged during the data extraction process, i.e., inductive coding. Table 10 shows the information extracted from the primary studies (i.e., data extraction forms) that we have used to define the codes.
- *Second cycle coding (pattern codes).* In this step, codes have been grouped into smaller number of categories, themes, or constructs (i.e., pattern codes). Pattern codes are explanatory or inferential codes that identify an emergent theme, configuration, or explanation [37]. In Section 2.4, the pattern codes of this SMS are described in the RQs where they have been identified.

The process has consisted of several iterations in which codes were added, modified, and removed over time in order to ensure the validity and consistency of the results.

Table 10
Data extracted from primary studies

Data item
Full reference
Year of publication
Source (conference, journal, workshop)
Type of publication (academy, industry)
First author's affiliation (organization and country)
Relation(s) with other primary studies of this SMS (references, references and extends, extends)
Term(s) used for referring to the data gathering activity adaptation
Definition(s), if any, of adaptive monitoring
Application domain(s), if any, where adaptive monitoring is applied
Type of main research contributions (algorithm(s), architecture) of the approach and its generalizability level (problem-specific, domain-specific, generic)
Approach purpose of adapting the monitoring system
Monitoring system's element(s) adaptation supported by the approach.
Approach adaptation process trigger(s).
Method(s), if any, used by the approach for analyzing relevant runtime data.
Method(s) used by the approach for (planning and) making the adaptation decision(s).
Type of adaptation decision enactment process supported by the approach (manual, semi-automatic, automatic)
Type of adaptation executed by the approach for adapting the monitoring system (structural, parameter)
Type of approach evaluation (experiment, industry use case), if any, and type of system in which the evaluation is performed

2.3.4. Visualization

In order to present the findings of the study, we have used different kind of methods (e.g., tables and charts) (see Section 2.4). The goal is to condense the major data for further analysis and to represent and present the conclusions. Table 11 presents the variables that have been tabulated and are used to answer the RQs.

2.3.5. Validity threats

For any empirical study the discussion of validity threats is of importance and is a quality criterion for study selection [35]. This section presents the aspects of the research process that might represent threats to validity and the actions performed to mitigate them. According to the recommendations by Petersen et al. [35], the types of validity threats that should be taken into account are: descriptive validity, theoretical validity, generalizability validity, interpretive validity and repeatability.

Table 11
Data tabulated per research question

Data	RQ
Terms related to adaptive monitoring	RQ1.1
Number of studies per term related to adaptive monitoring	RQ1.1
Year at which each term related to adaptive monitoring has been first and last used	RQ1.1
Sources of adaptive monitoring definitions	RQ1.2
Adaptive monitoring definitions	RQ1.2
Number of studies per year	RQ2.1
Number and percentage of studies per type of source and year	RQ2.2
Number and percentage of studies per type of publication and year	RQ2.3
Number and percentage of studies per continent and year	RQ2.4
Number of studies per country	RQ2.4
Studies per approach and research field	RQ2.5
Adaptive monitoring application domains	RQ2.5
Studies citation relation(s) with other studies of this SMS	RQ2.5
Number and percentage of approaches per type of contribution and year	RQ3.1
Number and percentage of approaches per generalizability level and year	RQ3.2
Number and percentage of approaches per type of adaptation purpose and year	RQ4.1
Number of approaches per combination of types of adaptation purposes (for most relevant combinations)	RQ4.1
Number of approaches per adaptation purpose (for most relevant types)	RQ4.1
Number and percentage of approaches per element adapted and year	RQ4.2
Number of approaches per combination of elements adapted (for most relevant combinations)	RQ4.2
Number and percentage of approaches per type of adaptation trigger and year	RQ4.3
Number of approaches per combination of types of triggers (for most relevant combinations)	RQ4.3
Number of approaches per adaptation trigger (for most important types)	RQ4.3
Number and percentage of approaches per analysis method and year	RQ4.4
Number of approaches per combination of analysis methods (for most relevant combinations)	RQ4.4
Number and percentage of approaches per decision-making method and year	RQ4.5
Number of approaches per combination of decision-making methods (for most relevant combinations)	RQ4.5
Number and percentage of approaches per type of enactment and year	RQ4.6
Number and percentage of approaches per type of adaptation executed and year	RQ4.7
Number and percentage of approaches per type of evaluation and year	RQ5.1
Number and percentage of approaches per type of system in which the evaluation is performed and year	RQ5.2

► Descriptive validity

Descriptive validity is the extent to which observations are described accurately and objectively [35]. In order to reduce this threat, we have designed a data extraction template for supporting the recording of data. The template tries to objectify the data extraction process. The different

template items, in the form of codes, are linked to specific parts of the primary studies, so they can be revisited when required, as it has been the case during the analysis. Constraining the extraction process exclusively to the data contained in the publication itself objectifies the observations; however, we must be aware that using this method, papers' classification accuracy may be affected in some cases. For instance, in this SMS, authors' affiliation data corresponds to the affiliation of authors at publication time. In this case, we are aware that results for the geographic and orientation (industry or academic) classifications may differ if for instance authors' affiliation data at submission time would be considered instead.

► Theoretical validity

Theoretical validity is determined by our ability of being able to capture what we intend to capture. Confounding factors such as biases and selection of subjects play an important role [35]. In order to reduce this threat, first, in the study identification process we have complemented the automatic search with backward snowballing of all studies. Then, since the selection process has mainly been conducted by the applicant (biases may appear), we have scheduled a set of periodic meetings for discussing and refining the final set of included and excluded papers.

This study has been conducted during 2017 and written during end of 2017 and beginning of 2018. Hence, only studies from 2016 and earlier have been included in the analysis, which implies that there is a risk that a recent paper may be missing. In spite of this limitation, we consider our sample of primary studies a good representation since a total of 110 studies, organized in different approaches proposed from different monitoring application domains, were identified (see Figure 10). Furthermore, different types of publication venues are well represented (see Figure 6). Finally, during the extraction process, codes have been created by the applicant what could also affect the validity of this task. In order to reduce this threat, the supervisors have assessed the extracted data. Though, given that this step involves human judgment, the threat cannot be eliminated [35].

► Generalizability validity

There are two types of generalizability validity, internal and external [35]. Given that the identified primary studies come from different monitoring application domains and research fields, we consider internal generalizability not a major threat of this SMS. Regarding the external generalizability, since the results of our SMS are within the scope of adaptive monitoring and we do not attempt to generalize conclusions beyond this scope, validity threats in this regard do not apply.

► Interpretive validity

Interpretive validity is achieved when the conclusions drawn are reasonable given the data, and hence maps to conclusion validity [35]. In order to reduce this threat, the experienced

supervisors have revised insights obtained by the applicant and discussed with her possible misunderstandings.

► Repeatability

The repeatability requires detailed reporting of the research process [35]. We have reported the process that we have followed for conducting our SMS and described the actions taken to reduce threats to validity. We have also helped repeatability by using existing guidelines for conducting the review.

2.4 Results of the review

In this section, we address the RQs introduced in Table 8. With this goal, we summarize the results obtained from the data extraction process. The data extracted from the studies and used to address the RQs is available at [50].

2.4.1. RQ1. What is adaptive monitoring?

► RQ1.1 – What are the terms related to the term "adaptive monitoring"?

In order to answer this question, we have performed a first cycle coding (see Section 2.3.3) using the In Vivo method defined by Miles et al. [37]. Next, we have categorized those codes into different groups. We were looking for other terms used by the researchers for referring to adaptive monitoring. We have found that 90 out of the 110 studies use other terms for referring to adaptive monitoring. We have grouped these different terms into 33 categories. In order to do that, we have identified terms that could be variants of a simpler term and put them into the same group (e.g., Monitoring Reconfiguration, Reconfigurable Monitors, Self-configuring Monitoring, Monitors Configuration, among other similar terms, can been grouped into a category named Monitor Configuration). Some of the terms could not be grouped with others; therefore, a category for each of them has been created.

In Figure 4, we present the most relevant categories (i.e., categories of terms mentioned in more than one study) ordered by the total number of studies that mentioned them (number in parenthesis). Categories composed by more than one term are marked with an asterisk. We have included a category named *Adaptive Monitor* in which we grouped terms such *Adaptive Monitoring* or *Adaptive Monitors*. Terms in the *Adaptive Monitor* category have been found in 41 out of the 110 studies of this SMS. Regarding the rest of terms, as it can be noticed, terms grouped into the *Monitor Configuration* category are the most mentioned (17 papers) followed by *Adaptive Sampling* (16 papers).

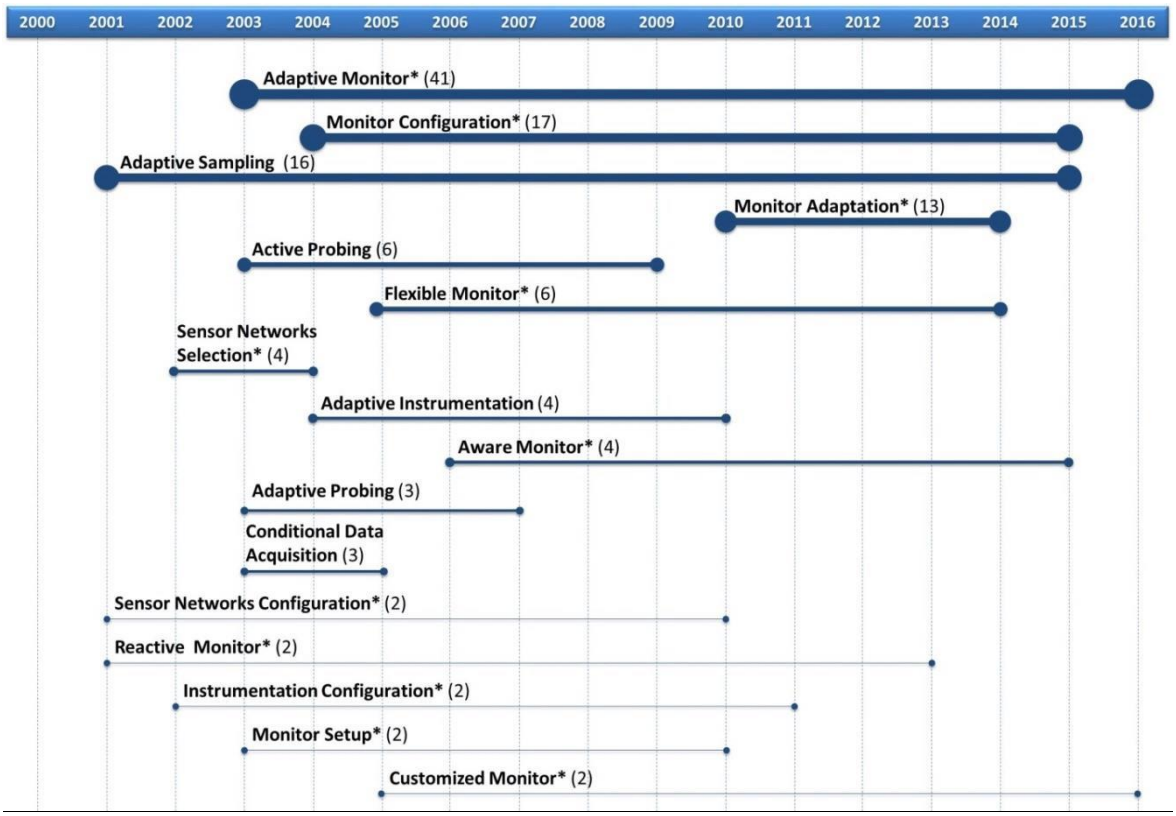


Figure 4: Categories of terms related to the term "adaptive monitoring" present in more than one study over the years

In this RQ, we were also interested on studying the way in which these terms have been used over the years. Thus, for each category, we have determined the year at which its terms have been, first and last used. Figure 4 shows how some groups of terms are well established in the community with a long life span (e.g., *Adaptive Monitor*, *Monitor Configuration* and *Adaptive Sampling*) while others show some obsolescence (e.g., *Active Probing*) or even a spurious momentum (e.g. *Conditional Data Acquisition*).

► RQ1.2 – Are there specific definitions of adaptive monitoring?

For answering this RQ, we have applied first cycle coding (see Section 2.3.3), restricted to the *Adaptive Monitor* category introduced in RQ1.1. As a result, we have identified that in the majority of the studies, there was no interest on defining the terms in the Adaptive Monitor category but instead on describing how they are actually realized (e.g., adjusting a variable, reconfiguring components). Specifically, we have found only 2 out of the 41 studies that actually present a definition for the term Adaptive Monitoring. Both works are from the same authors and the definition presented was the same as well. Concretely, authors define Adaptive Monitoring as:

> *"The ability an online monitoring function has to decide and to enforce, without disruption, the adjustment of its behavior for maintaining its effectiveness, in respect of the variations of both functional requirements and operational constraints, and possibly for improving its efficiency according to self-optimization objectives."*
>
> ***Moui et al.***, *A CIM-based Framework to Manage Monitoring Adaptability*[51], *Information Models for Managing Monitoring Adaptation Engorcement* [52]

2.4.2. RQ2. What are the demographic characteristics of the studies about adaptive monitoring?

► RQ2.1 - When are the studies published?

To answer this RQ, we have also applied only first cycle coding (see Section 2.3.3). Concretely, we have created a pre-defined list of codes deduced from the publication years we are considering in this SMS (2000 to 2016). Figure 5 shows the number of studies published per year.

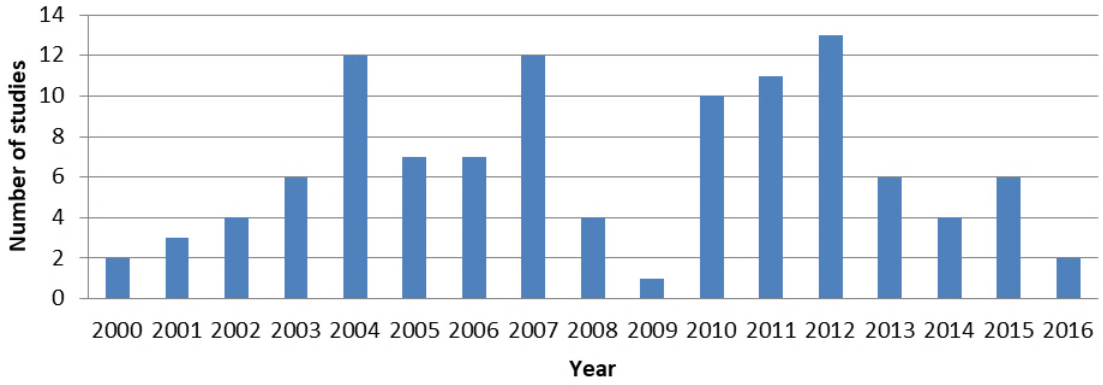
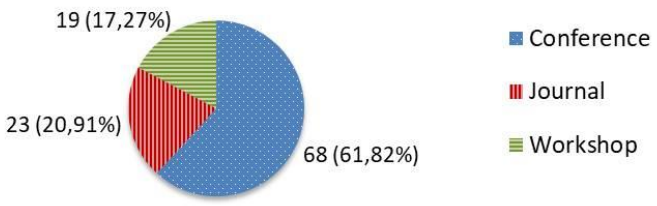


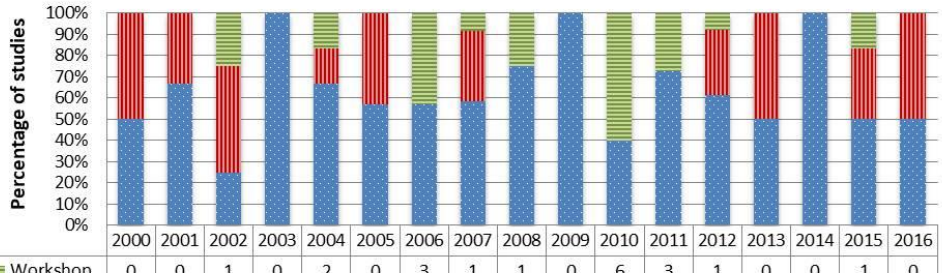
Figure 5: Number of studies published per year

RQ2.2 - Where are the studies published?

For addressing this RQ, we have conducted an inductive first cycle coding (see Section 2.3.3), using the In Vivo method [37] on the name of the sources. Then, we have classified the sources by type: *Conference*, *Journal,* and *Workshop*. The distribution of the 110 primary studies among these categories is shown in Figure 6a. According to our data, conference proceedings (with 68 papers) are the most prevalent publication type. Figure 6b shows the percentage of studies published in the different types of sources per year.



(a)



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Workshop	0	0	1	0	2	0	3	1	1	0	6	3	1	0	0	1	0
Journal	1	1	2	0	2	3	0	4	0	0	0	0	4	3	0	2	1
Conference	1	2	1	6	8	4	4	7	3	1	4	8	8	3	4	3	1

(b)

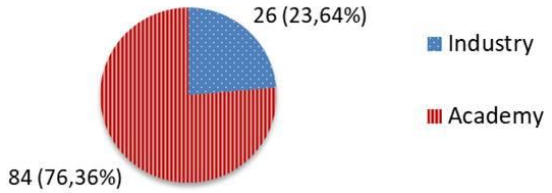
Figure 6: Number and percentage of studies per source type: (a) total, (b) over the years.

► RQ2.3 - How are publications distributed between academy and industry?

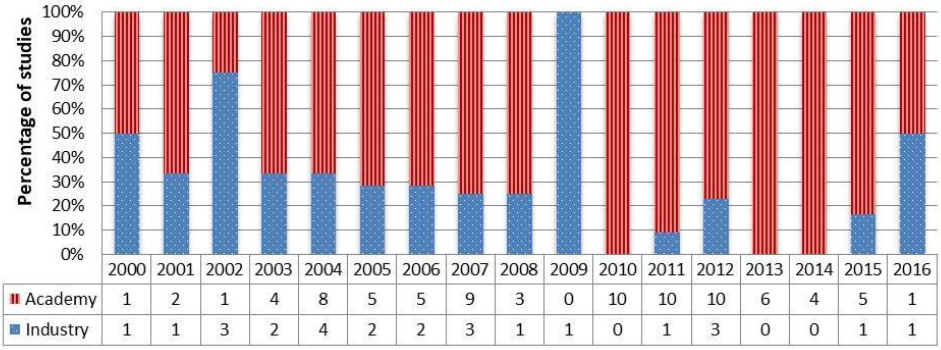
In order to answer this question, we have analyzed whether all authors of a study come from academic institutions (similarly to the approach applied by Franco-Bedoya et al. [53]), and applied first cycle coding (see Section 2.3.3). Figure 7a shows that 84 out of the 110 studies are from *Academy*, while 26 out of the 110 studies have at least on author from *Industry*. From the 26 studies coded as *Industry* publications, we have found that 12 studies are exclusively authored by researchers affiliated to industry. In Figure 7b, we provide an overview of the percentage of *Industry* and *Academy* studies published per year.

► RQ2.4 - How are publications geographically distributed?

In a SMS, the geographical distribution of the studies allows researchers to identify which continents (and countries) are making significant contributions to a specific research topic, and which are leading in terms of research publications [54]–[56]. In this SMS, the geographical data extracted from studies uncovers the locations of the main researchers interested on the adaptive monitoring topic. In order to do so, we have conducted a first cycle coding (see Section 2.3.3) using the In Vivo method [37] on the whole affiliation information of the first author of each study. Then, for the second cycle of coding (see Section 2.3.3), we have done two iterations: first, we have categorized affiliations per country; second, we have grouped countries by continents.



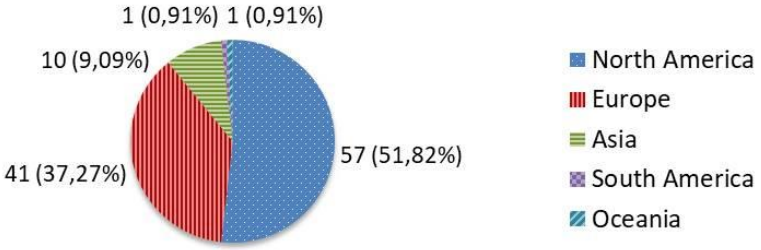
(a)



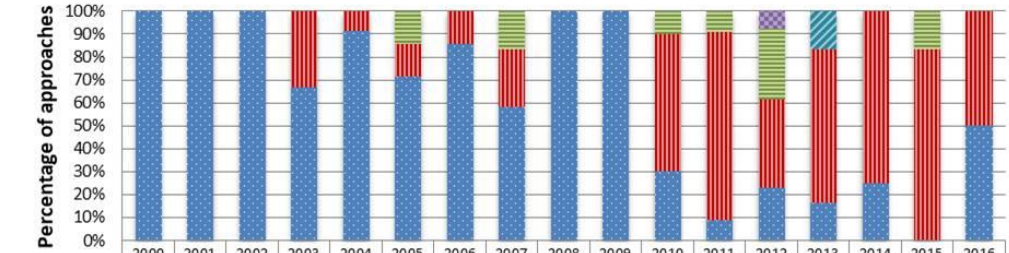
	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Academy	1	2	1	4	8	5	5	9	3	0	10	10	10	6	4	5	1
Industry	1	1	3	2	4	2	2	3	1	1	0	1	3	0	0	1	1

(b)

Figure 7: Number and percentage of industry and academy studies: (a) total, (b) over the years.



(a)



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Oceania	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
South America	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Asia	0	0	0	0	0	1	0	2	0	0	1	1	4	0	0	1	0
Europe	0	0	0	2	1	1	1	3	0	0	6	9	5	4	3	5	1
North America	2	3	4	4	11	5	6	7	4	1	3	1	3	1	1	0	1

(b)

Figure 8: Number and percentage of studies published per continent: (a) total, (b) over the years

In Figure 8a, we show the distribution of studies among the different continents. *North America* (57 papers) and *Europe* (41 papers) are the most dominant continents. Figure 8b provides information about the percentage of studies published in each continent by year. It can be noticed that until 2010, studies were mainly published by institutions placed in *North America*. Afterwards, *Europe* takes the lead. Finally, in Figure 9, we display how studies are geographically distributed among the different countries. *USA* is by far the country with more published studies (51 papers).

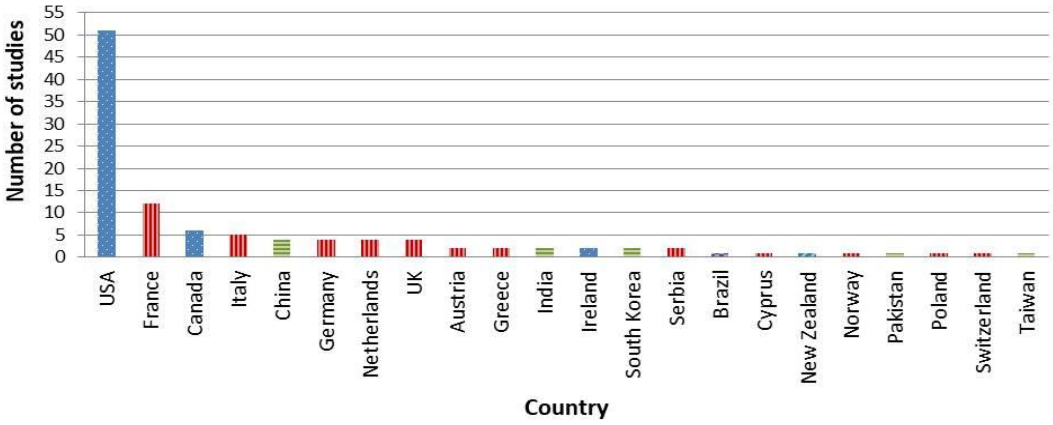


Figure 9: Number of studies published per country

► RQ2.5 – How are the studies organized into approaches for adaptive monitoring?

In order to organize the studies into approaches, we have determined, based on the list of authors and full-text reading of the articles, which studies were extended by other studies (i.e., belong to the same approach according to our interpretation). We have conducted a first cycle coding (see Section 2.3.3), creating a network of the 110 primary studies, using Atlas.ti® in which we indicate which studies reference and extend, or are extended by (but not referenced by), other studies. As a result, 81 approaches have been identified, 64 composed of only 1 study and 17 consisting of more than one. In Figure 10, we represent the 110 studies by small circles. We have assigned to each circle a resource identifier (extracted from the list of references provided in Appendix A1). The studies that are part of the same approach have been grouped into bigger circles (circles numbered from 1-17 in Figure 10).

During the analysis, we have also extracted the citation information among the studies of the SMS. In Figure 10, this information is shown in the form of arrows. Some of the studies had not citation relation with other studies of the systematic mapping. In Figure 10, these studies were grouped into the rectangle placed at the bottom of the figure. Once placed, we have classified the studies in different abstract topics or research fields that predominated on each cluster. The categories are shown in Figure 10 in the form of circles tagged with the topic or field name. The

rectangle containing the studies without citation relation has been tagged as *Various* (since those studies are cross-domain).

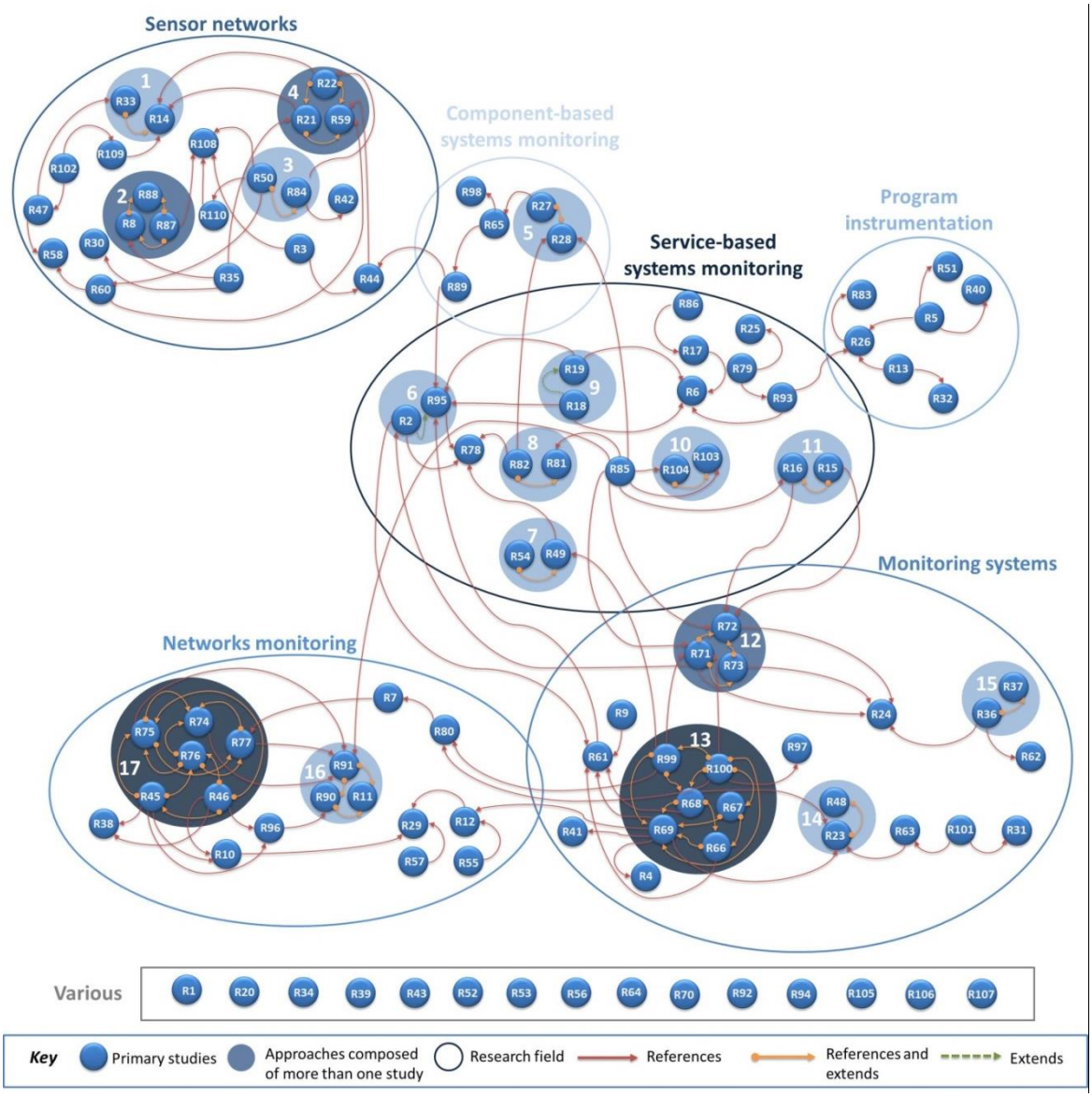


Figure 10: Studies organized by citing information in approaches and research fields

Finally, we have further analyzed the studies in order to identify the application domains where adaptive monitoring is applied. Not all the studies provide examples of applications and there are studies that provide more than one example. In order to extract the data from studies, we have conducted both types of coding (see Section 2.3.3). First, using the In Vivo method [37], we have coded all the application examples found in the studies. Then, we have grouped similar applications into application domains. In total, 27 categories for application domains have been identified (see Figure 11). In Figure 11, we have organized the domain categories by the number of application examples provided by studies (number in parenthesis). As it can be

noticed, the *Web applications monitoring* domain (from the Service-based systems field) is the most popular, followed by the *Object tracking* application domain (normally realized by sensor networks).

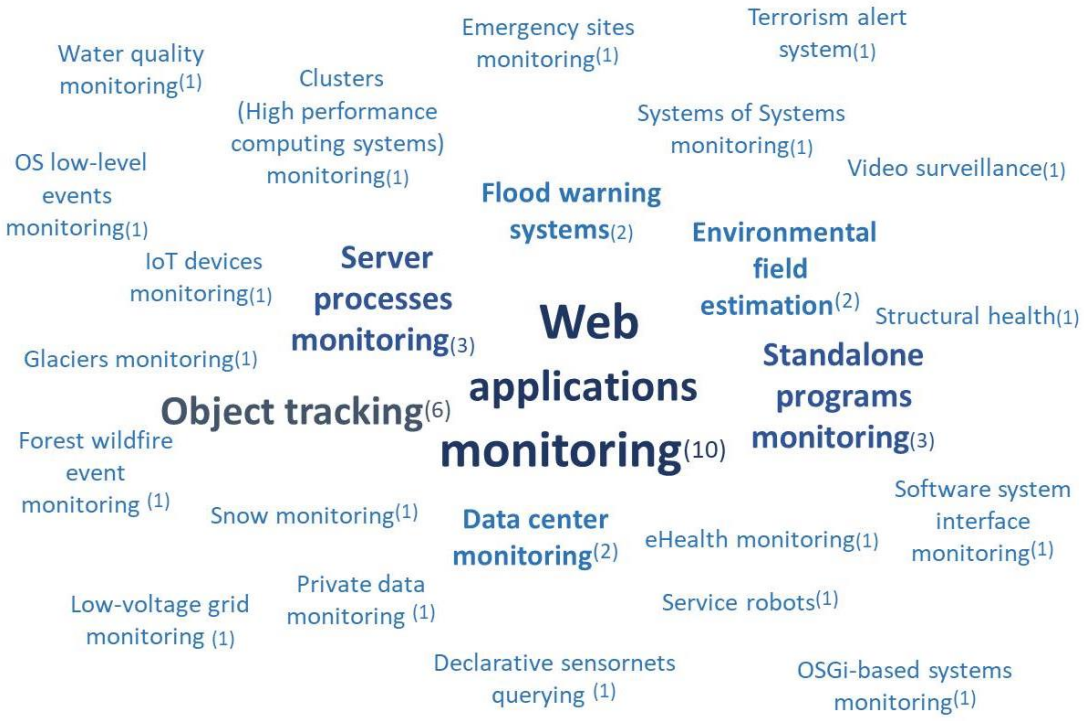


Figure 11: Application domains in which adaptive monitoring is applied by the studies

2.4.3. RQ3. What is proposed by adaptive monitoring approaches?

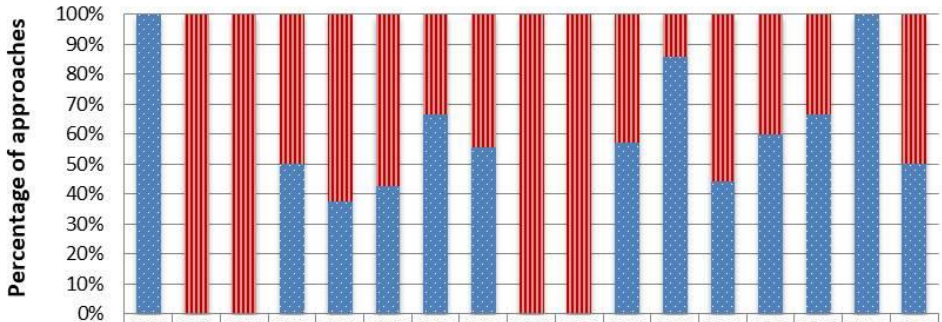
Given their more profound intent, the research questions RQ3, RQ4 and RQ5, have been analyzed considering the 81 approaches instead of the individual papers. For the 17 approaches composed by more than one study, we have mainly based the analysis on either the latest published study of the set or the most complete version (e.g., journal publications may provide more details than conference proceedings). Occasionally, we have revised other studies of the set to clarify unclear issues. It is worth remarking that, when visualizing the approaches by year, we have used the year of the last contribution, i.e., the study of the set with the latest publication date. Finally, in order to focus on trends when further exploring second cycle categories (when applicable), we calculate the average number of approaches per category in each research sub-question and focus on categories present in a total number of approaches above this average.

► RQ3.1 - What type of contributions are presented?

Based on the type of proposals presented by the studied approaches, we have derived the codes: *Algorithm(s)-only* and *Algorithm(s) and architecture* with which we have conducted a first cycle coding (see Section 2.3.3). Figure 12a shows that the contributions of 42 approaches are of the

type *Algorithm(s) and architecture* while the contributions of 39 approaches are *Algorithm(s)-only*. In Figure 12b, we condense the information about the percentage of published approaches per type of contribution over the years.

(a)



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Algorithm(s)-only	0	3	3	2	5	4	1	4	3	1	3	1	5	2	1	0	1
Algorithm(s) and architecture	1	0	0	2	3	3	2	5	0	0	4	6	4	3	2	6	1

(b)

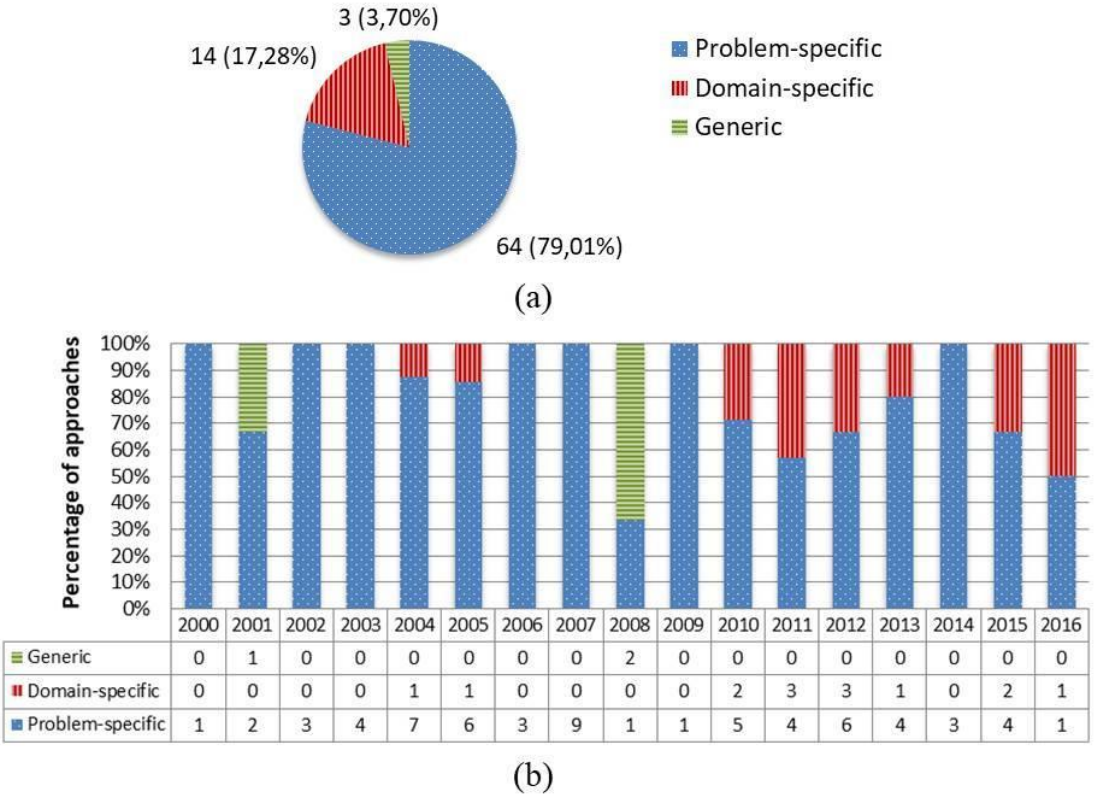
Figure 12: Number and percentage of approaches per type of contribution: (a) total, (b) over the years

► RQ3.2 - How generic are the solutions presented?

For answering this question, we have classified approaches' solutions in three main types: *Problem-specific, Domain-specific,* and *Generic*. *Problem-specific* solutions correspond to approaches that try to solve a specific problem in a specific domain, e.g., an algorithm for adapting the path of mobile sensors in order to improve monitoring precision when supervising water quality. *Domain-specific* solutions are considered for approaches supporting adaptive monitoring in a specific domain but without constraining the solution to a specific problem, e.g., an approach for supporting monitoring rules adaptation in WS-BPEL processes through dynamic weaving. Finally, the *Generic* category corresponds to solutions that can be applied in any domain, e.g., a threshold-based solution for changing monitoring systems' sampling rate.

We have conducted a first cycle coding (see Section 2.3.3), and as a result, we have found 64 approaches proposing *Problem-specific* solutions, 14 providing *Domain-specific* solutions and 3 presenting *Generic* ones (see Figure 13a). Figure 13b shows the percentage of approaches per

type of solution over the years. As it can be noticed, most of the *Domain-specific* solutions belong to approaches with contributions published after 2007.



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Generic	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
Domain-specific	0	0	0	0	1	1	0	0	0	0	2	3	3	1	0	2	1
Problem-specific	1	2	3	4	7	6	3	9	1	1	5	4	6	4	3	4	1

Figure 13: Number and percentage of approaches per type of solution: (a) total, (b) over the years

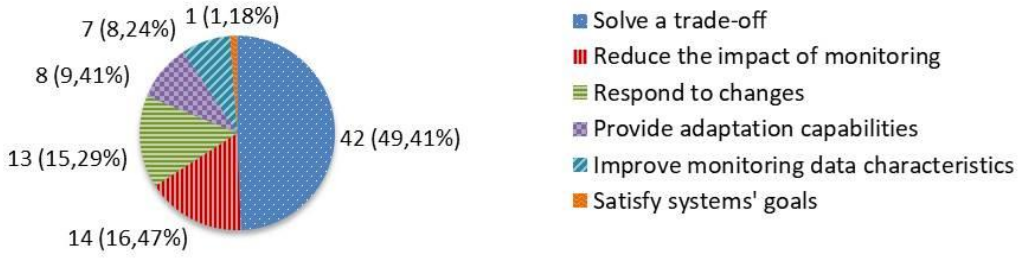
2.4.4. RQ4. How adaptive monitoring is conducted by the approaches?

► RQ4.1 - What is the purpose of adaptation?

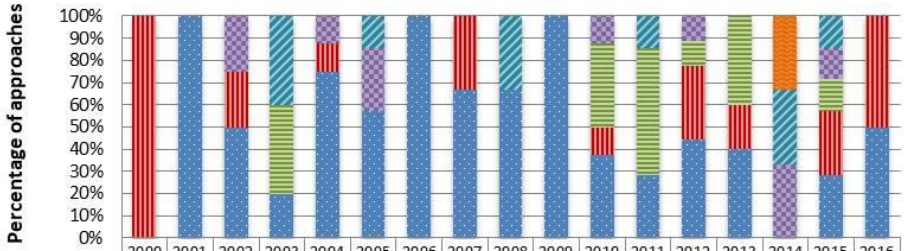
To answer this RQ, we have first derived from the approaches all the different adaptation purposes in the form of descriptive codes, i.e., inductive first cycle coding (see Section 2.3.3). Then, we have classified these purposes into different types. Figure 14a shows the number of approaches motivated by the different types of purposes. The most popular type is *Solve a trade-off* (42 approaches). There are some approaches motivated by two types of purposes; however, except for one pair of purposes that was used by two approaches (*Provide adaptation capabilities* and *Respond to changes*) each combination of purposes was used just by one approach. In Figure 14b, the percentage of approaches per type of purpose is displayed by year. This figure shows that *Solve a trade-off* has motivated approaches for a long timespan (from 2001 to 2016).

The average amount of approaches per type of purpose is 14,17. As it can be noticed, *Solve a trade-off* type of purpose is by far above this average, thus we further explore it. This type of purpose is composed of 14 different trade-offs, here we focus in the most relevant ones, i.e., trade-offs motivating more than one approach. From the most to the least popular:

- Improve the understanding about the SuM *while* Reducing the overhead associated with monitoring (14 approaches)
- Improve the understanding about the SuM *while* Reducing the energy consumption (9 approaches)
- Improve monitoring data accuracy *while* Reducing the overhead associated with monitoring (6 approaches)
- Improve monitoring data accuracy *while* Not exceeding available resources (2 approaches)
- Improve monitoring coverage *while* Reducing energy consumption (2 approaches)



(a)



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Satisfy systems' goals	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Improve monitoring data characteristics	0	0	0	2	0	1	0	0	1	0	0	1	0	0	1	1	0
Provide adaptation capabilities	0	0	1	0	1	2	0	0	0	0	1	0	1	0	1	1	0
Respond to changes	0	0	0	2	0	0	0	0	0	0	3	4	1	2	0	1	0
Reduce the impact of monitoring	1	0	1	0	1	0	0	3	0	0	1	0	3	1	0	2	1
Solve a trade-off	0	3	2	1	6	4	3	6	2	1	3	2	4	2	0	2	1

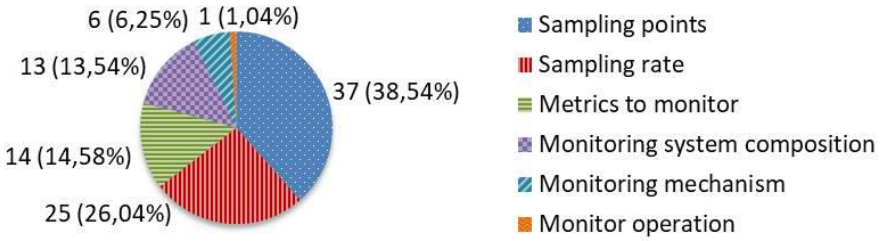
(b)

Figure 14: Number and percentage of approaches
per type of adaptation purpose: (a) total, (b) over the years

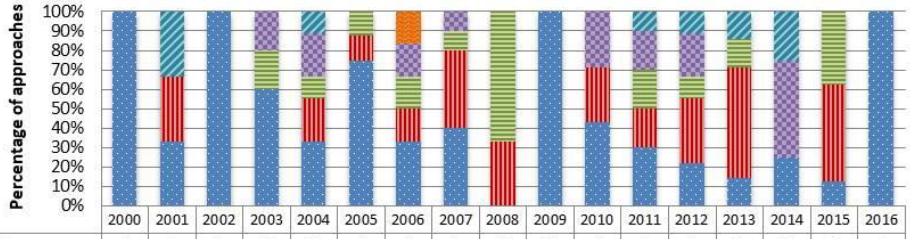
▶ RQ4.2 - What is adapted?

In order to address this question, we have derived codes that describe what is adapted by existing approaches during the data extraction process, i.e., we have conducted inductive first cycle coding (see Section 2.3.3). Figure 15a shows the aspects that existing approaches adapt and the number of approaches that support the adaptation of each aspect. In Figure 15b, we provide the percentage of approaches per year that support the adaptation of a specific aspect. As it can be noticed, the most adapted aspects are the *Sampling points* (37 approaches) and the *Sampling rate* (25 approaches). Moreover, the relevance of the adaptation of these aspects over the years is evident, particularly for the *Sampling points* (present from 2000 to 2016 except for 2008).

Some of the approaches support the adaptation of more than one aspect. From the most to the least popular, the most relevant combinations of elements supported by existing approaches, i.e., combinations supported by more than one approach, are: *Metrics to monitor* and *Sampling points* (4 approaches) and *Metrics to monitor* and *Sampling rate* (2 approaches).



(a)



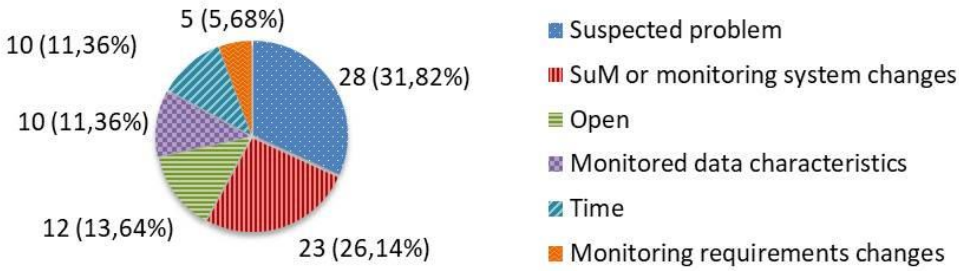
	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Monitoring operation	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Monitoring mechanism	0	1	0	0	1	0	0	0	0	0	0	1	1	1	1	0	0
Monitoring system composition	0	0	0	1	2	0	1	1	0	0	2	2	2	0	2	0	0
Metrics to monitor	0	0	0	1	1	1	1	1	2	0	0	2	1	1	0	3	0
Sampling rate	0	1	0	0	2	1	1	4	1	0	2	2	3	4	0	4	0
Sampling points	1	1	3	3	3	6	2	4	0	1	3	3	2	1	1	1	2

(b)

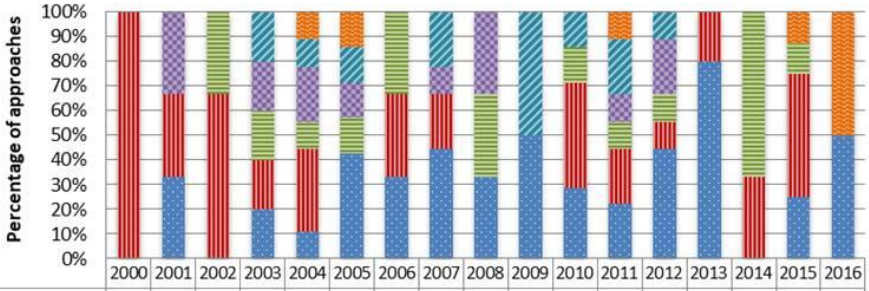
Figure 15: Number and percentage of approaches per element adapted: (a) total, (b) over the years

► RQ4.3 - What triggers adaptation?

For answering this question, we have applied both cycles of coding (Section 2.3.3). First, we have derived a set of codes for describing the different triggers we have found in existing approaches. Then, we have grouped them by type. In Figure 16a, the number of approaches per type of trigger is presented while Figure 16b shows the percentage of approaches per trigger type over the years. A *Suspected problem* is the most common factor that triggers adaptation in existing approaches (28 approaches); the relevance of this type of trigger is corroborated by its long and continuous presence in approaches over the years (from 2001 to 2016 with just two years of absence, 2002 and 2014). Some of the approaches use more than one type of factor for triggering the monitoring adaptation process. The most relevant combinations of types of triggers we have found, i.e., combinations used by more than one approach, are: *Suspected problem* and *Time* (2 approaches) and *SuM or monitoring system changes* and *Monitoring requirements changes* (2 approaches).



(a)



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Monitoring requirements changes	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1
Time	0	0	0	1	1	1	0	2	0	1	1	2	1	0	0	0	0
Monitored data characteristics	0	1	0	1	2	1	0	1	1	0	0	1	2	0	0	0	0
Open	0	0	1	1	1	1	1	0	1	0	1	1	1	0	2	1	0
Monitored or monitoring system changes	1	1	2	1	3	0	1	2	0	0	3	2	1	1	1	4	0
Suspected problem	0	1	0	1	1	3	1	4	1	1	2	2	4	4	0	2	1

(b)

Figure 16: Number and percentage of approaches per type of adaptation trigger: (a) total, (b) over the years

According to data shown in Figure 16a, the average amount of approaches per type of trigger is 14,67. Thus, we further explore the *Suspected problem* and *SuM or monitoring system changes*

types. *Suspect problem* type of trigger is composed of 7 triggers, the most relevant, i.e., triggers present in more than one approach, from the most to the least popular:

- Monitoring system component anomaly (11 approaches)
- Requirement or constraint violation (5 approaches)
- Requirement or constraint likely to be violated (4 approaches)
- SuM component anomaly (3 approaches)
- SuM component likely to present an anomaly (2 approaches)
- Likely environmental problem (2 approaches)

Anomalies in systems' components include faults. On the other hand, the *SuM or monitoring system changes* category is composed of 4 triggers, from most to less popular:

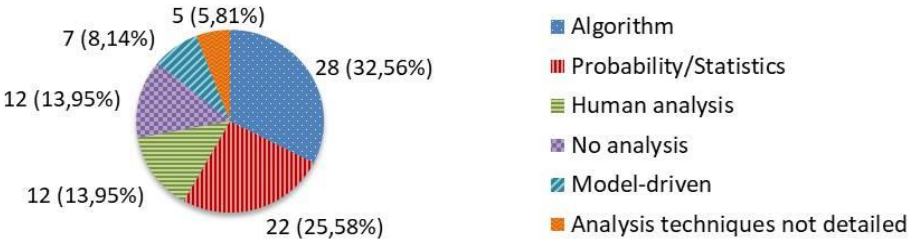
- SuM state changes (16 approaches)
- SuM components de/activation (4 approaches)
- Monitoring system components addition/removal (2 approaches)
- Execution context changes (2 approaches)

Thus, in conclusion a change in the SuM state is the most popular trigger.

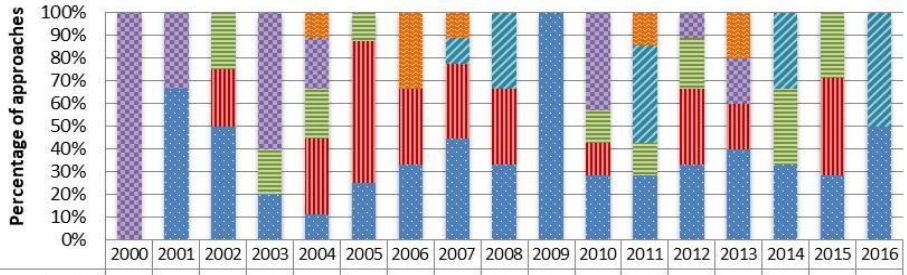
► RQ4.4 - How is analysis performed?

To answer this question, we have conducted an inductive first cycle coding for identifying analysis solutions, and then a second cycle coding for grouping them by type (see Section 2.3.3). Categories for grouping approaches that do not perform analysis or do not provide details about how it is performed have also been created. Figure 17a shows the categories created as well as the number of approaches per category. As it can be noticed, most of the approaches use a specially designed *Algorithm* for conducting the analysis task (28 approaches) followed by solutions that use *Probability/Statistics* (22 approaches).

Figure 17b provides the details about the percentage of approaches using specific types of analysis per year. In this figure, the relevance of the *Algorithm* category is corroborated since this type of analysis is present every year from 2001 to 2016. During the data extraction process, we have found that this type of analysis is combined with *Probability/Statistics* by two approaches. Other combinations, e.g., *Human analysis* and *Probability/Statistics*, have been also identified; however, since they were used only by one approach each, they have not been considered relevant for the purposes of this SMS (i.e., finding trends). For the same reason, we have no further decomposed the most relevant categories (i.e., *Algorithm* and *Probability/Statistics*); every analysis solution in these categories is unique which do not provide information relevant for finding trends.



(a)



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Analysis techniques not detailed	0	0	0	0	1	0	1	1	0	0	0	1	0	1	0	0	0
Model-driven	0	0	0	0	0	0	0	1	1	0	0	3	0	0	1	0	1
No analysis	1	1	0	3	2	0	0	0	0	0	3	0	1	1	0	0	0
Human analysis	0	0	1	1	2	1	0	0	0	0	1	1	2	0	1	2	0
Probability/Statistics	0	0	1	0	3	5	1	3	1	0	1	0	3	1	0	3	0
Algorithm	0	2	2	1	1	2	1	4	1	1	2	2	3	2	1	2	1

(b)

Figure 17: Number and percentage of approaches per analysis type: (a) total, (b) over the years

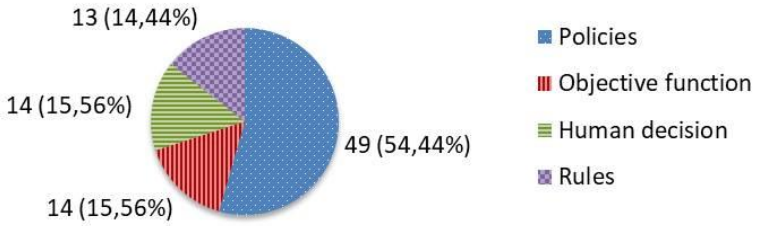
► R4.5 - How adaptation decisions are made?

For addressing this question, we have derived codes based on the type of criterion used by existing approaches for making adaptation decisions, i.e., inductive first cycle coding (see Section 2.3.3). Resulting codes are shown in Figure 18a. *Polices* is the most used type of criterion for conducting the decision-making process in existing approaches (49 approaches). In Figure 18b, we provide an overview of the percentage of approaches using the different types of decision-making criteria over the years. This figure show clearly that *Policies* have played an important role in decision-making processes since, apart from being the most used type of criterion, they have been utilized by approaches since 2000 till 2016 (except for 2009). *Policies* have also been combined in existing approaches with the other types of decision-making criteria. Concretely, four approaches have combined them with *Human decision*, three with *Rules,* and two with an *Objective function*. We have not found any combination that does not involve *Policies*.

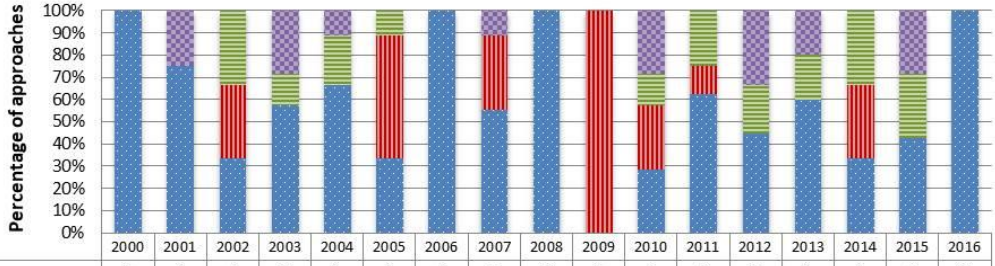
► RQ4.6 - How adaptation decisions are enacted in the monitoring system?

Three codes for describing the type of enactment process have been derived from existing approaches in order to answer this question: *Automatic*, *Semi-automatic,* and *Manual* (see Figure 19). *Automatic* enactment has been assigned to the approaches that perform the

adaptation of the monitoring system without any human intervention. *Semi-automatic* is assigned to the approaches that require human intervention at some degree, for instance, approaches requesting human approval before enacting adaptations. Finally, *Manual* enactment corresponds to approaches in which the enactment of the adaptations is completely performed by humans.



(a)



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Rules	0	1	0	2	1	0	0	1	0	0	2	0	3	1	0	2	0
Human decision	0	0	1	1	2	1	0	0	0	0	1	2	2	1	1	2	0
Objective function	0	0	1	0	0	5	0	3	0	1	2	1	0	0	1	0	0
Policies	1	3	1	4	6	3	3	5	3	0	2	5	4	3	1	3	2

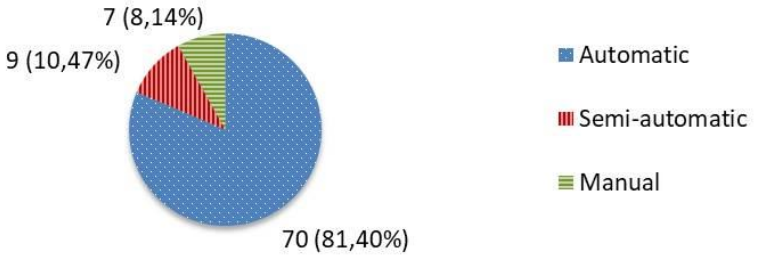
(b)

Figure 18: Number and percentage of approaches per decision-making type: (a) total, (b) over the years

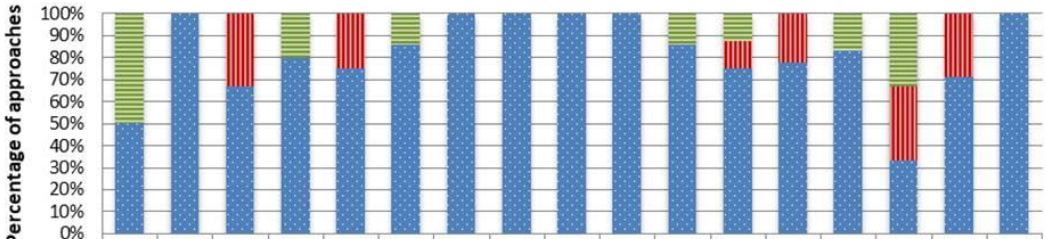
Figure 19a shows the distribution of approaches among the different types, resulting from a first cycle coding (see Section 2.3.3). The percentage of approaches using the different types of enactment per year is shown in Figure 19b. *Automatic* is by far the type of enactment most used by existing approaches (70 approaches published between 2000 and 2016). During the data extraction, we have identified four approaches that support both *Automatic* and *Manual* enactment.

► RQ4.7 - What type of adaptation is executed?

For addressing this RQ, we have considered two codes that describe two different types of adaptation: *Structural* and *Parameter* (see Figure 20). The first one refers to changes in the structure of the monitoring system, such as the exchange of components or a new composition of components [2]. The second one refers to changes in the monitoring system's parameters, such as the change of the sampling rate or the change of the list of metrics to monitor [2]. Using these codes, we have conducted a first cycle coding (see Section 2.3.3).



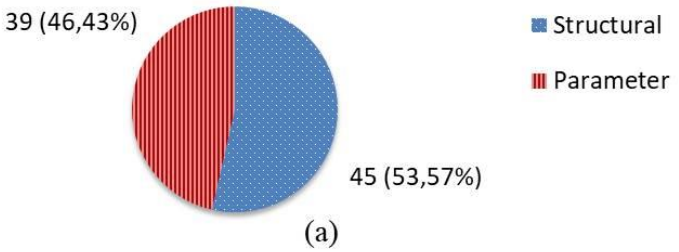
(a)



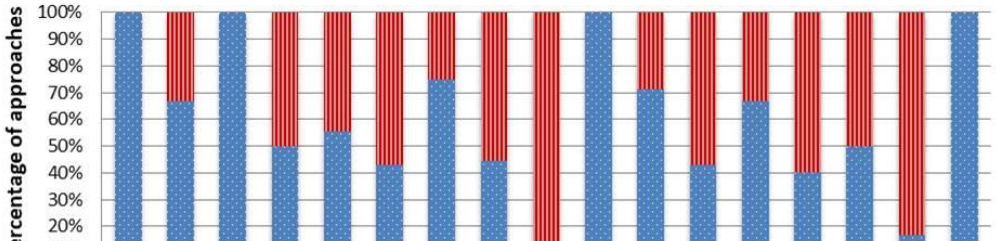
	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Manual	1	0	0	1	0	1	0	0	0	0	1	1	0	1	1	0	0
Semi-automatic	0	0	1	0	2	0	0	0	0	0	0	1	2	0	1	2	0
Automatic	1	3	2	4	6	6	3	9	3	1	6	6	7	5	1	5	2

(b)

Figure 19: Number and percentage of approaches per enactment type: (a) total, (b) over the years



(a)



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Parameter	0	1	0	2	4	4	1	5	3	0	2	4	3	3	2	5	0
Structural	1	2	3	2	5	3	3	4	0	1	5	3	6	2	2	1	2

(b)

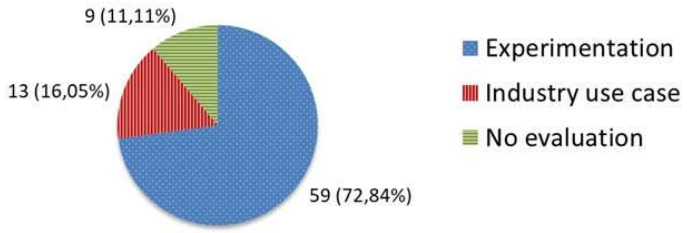
Figure 20: Number and percentage of approaches per type of adaptation executed: (a) total, (b) over the years

Figure 20a shows that in existing approaches most of the adaptation decisions have been translated into *Structural* monitoring systems' changes (45 approaches). From the extracted data, we have identified three approaches that support both types of adaptations. In Figure 20b, we provide an overview of the percentage of approaches per type of adaptation over the years.

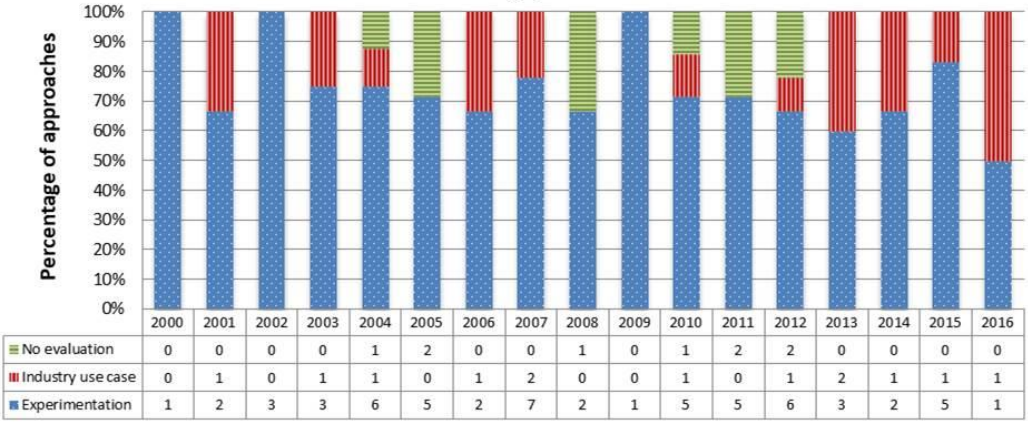
2.4.5. RQ5. How adaptive monitoring approaches are evaluated?

► RQ5.1 - What type of evaluation is performed?

To address this question, we have derived a set of codes based on the types of evaluation we have found in existing approaches, if any, and conducted a first cycle coding (see Section 2.3.3). The resulting codes are: *Experimentation*, *Industry use case* and *No evaluation* (see Figure 21). We have assigned the *Experimentation* code to approaches conducting their evaluation in simulated systems. The *Industry use case* code has been assigned to approaches conducting their evaluation in real systems, both in controlled and production environments. Approaches presenting theoretical examples or no evaluation, where grouped into the *No evaluation* category.



(a)



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
No evaluation	0	0	0	0	1	2	0	0	1	0	1	2	2	0	0	0	0
Industry use case	0	1	0	1	1	0	1	2	0	0	1	0	1	2	1	1	1
Experimentation	1	2	3	3	6	5	2	7	2	1	5	5	6	3	2	5	1

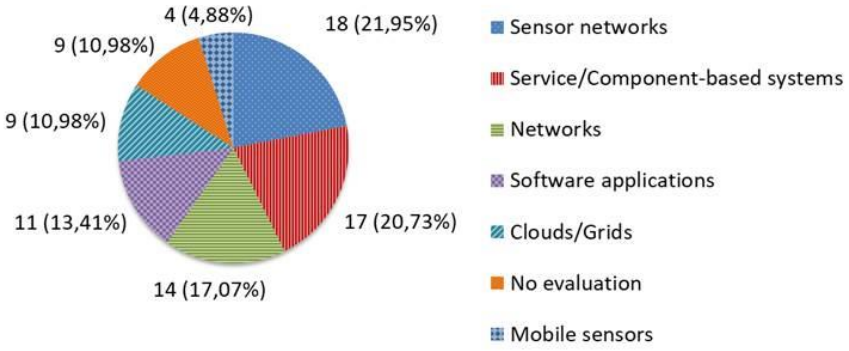
(b)

Figure 21: Number and percentage of approaches per evaluation type:
(a) total, (b) over the years

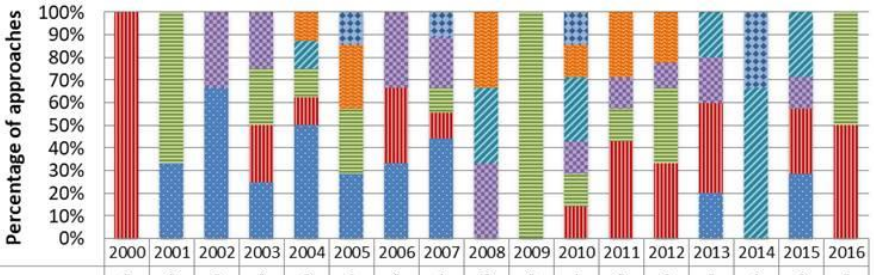
In Figure 21a, we provide the information about the number of approaches per type of evaluation. *Experimentation* has been the most used method for evaluating existing approaches (59 approaches). Information about how the different types of evaluation have been used over the years by existing approaches is displayed in Figure 21b. *Experimentation* and *Industry use case* have been present over long timespans (2000 to 2016 for *Experimentation* and 2001 to 2016 for *Industry use case*). We have not found approaches utilizing more than one type of evaluation.

► RQ5.2 – In which type of systems is the evaluation performed?

For answering this question, codes describing the different types of systems in which approaches are evaluated have been progressively added during the data extraction process, i.e., we have applied inductive first cycle coding (see Section 2.3.3). Figure 22a shows the number of approaches per system type (approaches not evaluated have been grouped into a *No evaluation* code).



(a)



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Mobile sensors	0	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0
No evaluation	0	0	0	0	1	2	0	0	1	0	1	2	2	0	0	0	0
Clouds/Grids	0	0	0	0	1	0	0	0	1	0	2	0	0	1	2	2	0
Software applications	0	0	1	1	0	0	1	2	1	0	1	1	1	1	0	1	0
Networks	0	2	0	1	1	2	0	1	0	1	1	1	3	0	0	0	1
Service/Component-based systems	1	0	0	1	1	0	1	1	0	0	1	3	3	2	0	2	1
Static sensor networks	0	1	2	1	4	2	1	4	0	0	0	0	0	1	0	2	0

(b)

Figure 22: Number and percentage of approaches per type of system in which the evaluation is performed: (a) total, (b) over the years

The most common types are: *Sensor networks* (composed of non-mobile sensors) present in 18 approaches, *Service/component-based systems* utilized by 17 approaches and *Networks* used for evaluating 14 approaches. Figure 22b shows the percentage of approaches evaluated in a specific type of system per year. According to this figure, most of the approaches evaluated in *Sensor networks* were published before 2008, while a wave of approaches performing evaluation in *Service/component-based systems* has been experienced after 2009. Evaluation in *Networks* cannot be characterized based on this figure. We have identified only one approach that has been evaluated in more than one type of system (*Sensor networks* and *Clouds/Grids*).

2.5 Discussion

In this section, we apply Data Mining techniques to the resulting codes of Section 2.4 in order to find further insights about the current state of the art of adaptive monitoring approaches. We are interested on identifying patterns in the approaches that cannot be easily determined by traditional analysis techniques, such the ones used in Section 2.4. Moreover, we analyze the results and discuss our findings for each research question.

2.5.1. Data Mining

Data Mining refers to the process of applying Machine Learning algorithms to data sets in order to discover patterns within the data. It is useful when human analysis is not feasible (e.g., very large amounts of data or high-dimensional data) and/or patterns are non-obvious. In literature reviews, Data Mining has been applied, for instance, in the form of text mining for supporting the study selection process [57]–[60]. In this SMS, we use Data Mining techniques for identifying patterns among the demographic characteristics of existing approaches (RQ2 of this SMS), the ways they present and conduct adaptive monitoring (RQ3, RQ4 of this SMS) as well as the evaluation processes (RQ5 of this SMS).

As we have mentioned before, in order to perform the Data Mining analysis, we have defined a set of variables based on the codes extracted in Section 2.4. The complete list of variables used is provided in Appendix A2 (Table A2-1). For conducting the analysis, we have used the rule-based JRip algorithm [38], [39] and the Waikato Machine Learning tool (Weka) [40]. In order to select the Data Mining algorithm, we have considered three main factors:

- ***Type of data***. The data to be mined are the codes resulting from the answers to this SMS RQs. That is, for each RQ there is a list of discrete possible values. This type of data is known as nominal data and the most intuitive method to mine this kind of data is *classification*.
- ***Algorithm complexity and results comprehensibility***. Among the available classifiers in Weka, we can find *networks*, *decision trees,* and *rule-based* classifiers. Networks are very well known by their complexity in terms of both computation and time required. Moreover, their classification results are less comprehensible (from a human point of

view) given the increased complexity of their models. Decision trees, on the other hand, are simpler classifiers that provide easy to comprehend classification results (in the form of trees). However, decision trees should be further processed in order to summarize the relations between variables in a more descriptive way (which is required in our case). Finally, rule-based classifiers, such the JRip, are very simple classifiers that provide easy to comprehend and easy to present classification results (in the form of descriptive rules) [61].

– ***Experience***. In some works that will be presented later in this thesis [7], [22], we have applied the rule-based JRip algorithm for different purposes. From our experience, this algorithm performs well with small data sets and the resulting classification rules are easy for us to read, understand, and trace to the mined data set.

We have run a classification for each variable, using the variable in turn as the class attribute of that run and the rest of variables as predictors. All resulting classifiers were evaluated using stratified 10-fold cross validation. The performance metrics produced for each classifier in Weka and that are averaged using the cross validation, include *precision*, *recall* and *f-measure* [40]. From each run, we have tabulated the classifier's resulting rules and the values of the performance metrics mentioned before.

In Appendix A2 (see Figure A2-1), we provide an overview of the performance metrics' values obtained for the classifier of each variable. The closer the values are to 1.0, the better the performance of the classifier. The criteria for deciding whether a classifier is good enough depend on the specific use case. In our case, we have not precedents for establishing criteria since we have not found any other review applying Data Mining to RQs answers for finding patterns. Thus, we have decided to consider classifiers with precision, recall, and f-measure greater than or equal to 0.9. As a result, classifiers for 17 out of the 47 analyzed variables were considered relevant. In Appendix A2 (Table A2-2), the list of rules that compose these 17 relevant classifiers is provided. In the rest of this section, we discuss the results we have obtained in Section 2.4 and complement the analysis with the cross-question patterns found.

2.5.2. Analysis of results

► RQ1. What is adaptive monitoring?

The diversity of research fields from which studies of our SMS have emerged, has certainly contributed to the diversification of the vocabulary used for referring to adaptive monitoring. This phenomenon can be clearly seen in Figure 4, in which we have presented the different terms categories utilized by 81,81% of the studies as alternatives to the term "adaptive monitoring". Most of these terms are domain-specific and in consequence cannot be reutilized in all the research fields.

One of the objectives of this study was to find a generic definition for the term "adaptive monitoring". In Section 2.4, we have looked for definitions in the studies of our SMS. As a result, we have found only one definition. Moreover, this definition is not complete and generic enough for being applied to the different realizations of adaptive monitoring we have found in this review, e.g., monitoring systems composition's adaptation based on SuM state changes, could not be covered with this definition.

In order to achieve our objective, we have adjusted the definition proposed by Moui et al. [51], [52] and created a generic definition for the term "adaptive monitoring":

> *"Adaptive monitoring is the ability a monitoring system has to modify its structure and/or behavior in order to respond to internal and external stimuli such changes in their execution context, functional and non-functional requirements, systems under monitoring or the monitoring system itself"*
>
> ***Zavala et al.**, Adaptive Monitoring: A Systematic Mapping* [18]

In this definition, all monitoring systems are treated equally (sensor networks, component-based software monitoring systems, instrumentation systems, etc.) which is beneficial for later standardizing other concepts applicable to all high-level monitoring systems as well, e.g., monitoring frequency adaptation. Moreover, unlike the definition proposed by Moui et al. [51], [52] which only consider the adjustment of behavioral aspects, in this definition adaptation is understood as changes in the monitoring system behavior as well as in its structure. Finally, the possible triggers of the adaptation process are not constrained in our definition, as they are in the one provided by Moui et al.

► RQ2. What are the demographic characteristics of the studies about adaptive monitoring?

The adaptation of monitoring systems is a lively research area with studies published every year from 2000 to 2016 (see Figure 5). However, it is remarkable that, if we take 3-year windows, the last period (2014-2016) is the one with fewer contributions (excluding the first period 2000-2002, when the topic was formulated). Interpretation of these trends needs always to be careful. On the one hand, the third period with fewer contributions was 2008-2010 but only a 2-year shift (2010-2012) yields to the most populated window. On the other hand, the advent of domains like Internet of Things (IoT), smart vehicles, etc., where self-adaptation and in relation to it, adaptive monitoring, is crucial, it may be expected a growth of contributions.

In terms of venue, most of the published papers in this topic have appeared in conference proceedings (see Figure 6); the percentage is very close to the average of 25.9% reported by Ameller et al. [62] from a sample of 14 systematic mappings in software engineering. On the other hand, we have not found any dominant venue in any of the categories. For instance, the conference with more publications is the International Conference on Network and Service

Management (CNSM) with 4 out of the 68 conference papers. In our opinion, this situation is due to the diversity of research fields in which adaptive monitoring is present. That is, papers are mainly published in venues specialized in the research field they belong to. This fact contributes to the isolation of solutions per research field and in some cases even per research communities. The need of venues in which the adaptive monitoring topic is central per se and research from different fields could be found and compared would help to promote this area.

From the type of publication perspective (see Figure 7), it is not surprising that the majority of the papers are from *Academy* (academics usually are more motivated to submit papers to conferences and journals [53]); however, the number of papers with authors affiliated to *Industry* indicates that adaptive monitoring is a topic of interest of practitioners as well. Moreover, this interest has been present almost every year considered in this review (see Figure 7b).

In this RQ, we have also explored how studies are distributed geographically. We have found that authors of North American and European organizations are the most active researchers in the adaptive monitoring topic (see Figure 8). This phenomenon could be explained by the numerous grants and research and innovation programs, often mentioned in the acknowledgements of the studies (e.g., the National Science Foundation and the European Community's 7th Framework Programme), funded by different organizations in these geographical areas. Regarding the geo-temporal distribution, from 2010 publications from North American organizations have dramatically decreased, 82,46% of their studies have been published before 2010 (see Figure 8b). The opposite effect has happened to European organizations' amount of publications. From this observation, we expect more European contributions in the next years in this topic than North American. Even though, *USA,* which is the main contributor in *North America,* is by far the country with most published papers (see Figure 9). Although this is the usual situation in Computing Science as reported by Ruiz [63], the difference is much greater (26.4% of the publications are from USA in this SMS). We do not expect others countries to reach the same amount of publications in the short-term.

Finally, we have analyzed how studies are organized in approaches. As a result, we have identified many different approaches (see Figure 10). The most prominent ones (approaches with more contributions) are from the *Networks monitoring* and the *Monitoring systems* (in general) research fields. While, studies composing the approach for *Networks monitoring* do not have interaction with studies of other fields (derived from citation data), the studies of the approach for supporting adaptation of *Monitoring systems* (in general) interact with studies of *Networks monitoring* as well as *Service-based systems monitoring* fields. This can be explained by the difference in the scopes of the approaches, i.e., while the first one tries to improve networks monitoring through adaptation, the second one aims at enabling adaptation capabilities in any type of monitoring system (e.g., networks or service-based systems).

In general, studies do not tend to reference studies of others research fields. The lack of interaction between the research communities, shown above in the venues' analysis, could be

the cause of this phenomenon. For finalizing the demographic analysis of studies, we have identified the application-domains where adaptive monitoring is applied. As we have foreseen, the adaptive monitoring topic has a wide range of applications (27 were found in the studies of this review). Regarding the application of Data Mining in the codes of this question, no relevant classifiers have been found.

► RQ3. What is proposed by the approaches?

The distribution of approaches between the two types of contributions identified is quite even (see Figure 12). However, it can be noticed in Figure 12b that approaches that present algorithms supported by architectural proposals have been more and more proposed in the last years. The opposite happens to approaches contributing with only algorithms. This phenomenon could be explained by the increasing need, in the last years, of monitoring systems' owners to provide formal solutions to the adaptation problem in order to support adaptive monitoring in complex domains such cloud-based applications, smart cities, etc. where isolated algorithms are not enough. Regarding the generalizability level of the solutions (see Figure 13), the majority is *Problem-specific* and cannot be reutilized or extended for dealing with other issues or supporting other adaptation functionalities. However, this type of solutions in some cases could be aggregated for instance, in order to solve a group of problems in a specific domain.

A concrete example of aggregation could be an approach that combines the context-aware e-health monitoring proposal of Mshali et al. [32] with a re-configurable service-based monitoring system infrastructure (e.g., Villegas et al. [64]) for supporting an energy-efficient monitoring system that can incorporate new sensors at runtime. Regarding the few *Generic* solutions we have found, unfortunately, they are all algorithmic solutions, not complete enough for providing a unified software engineering solution to current adaptive monitoring systems of the different research fields. The predominance of problem-specific solutions over generic ones can be explained by the lack of visibility of the adaptive monitoring contributions. Finally, when applying Data Mining to the codes derived in this RQ, no relevant classifiers have been found.

► RQ4. How adaptive monitoring is conducted by the approaches?

Unlike in previous RQs, relevant classifiers have been found for at least one code of each research sub-question of this SMS RQ (see Table A2-2 in Appendix A2). As we have mentioned before, classifiers are cross-question, this means that the resulting rules relate answers of one sub-question with the answers of the rest of sub-questions (including sub-questions of other RQs of this SMS). This allows us to identify cross-findings above the different RQs. For research sub-questions where answers' codes are not exclusive, e.g., in RQ4.1 more than one purpose could motivate an approach, relations between codes of the same sub-question were also mined.

The first rule found has been for the adaptation purposes, we have found that *Satisfying system's goals* is a purpose very unlikely to find in existing approaches (pattern that can also be visualized in Figure 14). We have also found that *Solve a trade-off* purpose do not usually motivate approaches in conjunction with other purposes (pattern expectable since in case of appearing in the same approach, purposes would form part of the trade-off). Apart from the patterns, we have noticed that after 2009 the variety of purposes considered by approaches has increased (see Figure 14b). This can be explained by the increasing diversity of applications, users' needs, and execution contexts identified for software systems, in the last years. We expect more and more varied factors motivating adaptive monitoring to emerge in the short-term.

Regarding what is adapted, a classifier confirming what is shown in Figure 15, regarding the unlikeliness of finding an approach adapting the monitor operation, has resulted. We have also found a classifier that positively relates *Monitoring system composition* adaptation to *Structural* changes and negatively the last one with *Sampling rate* adaptation. This makes sense since one can expect *Structural* changes when re-composition is required and a *Parameter* changes when the adaptation is of a variable such the *Sampling rate*. For the adaptation triggers, we have found a classifier that corroborates that *SuM or monitoring system changes* type of trigger is not combined with other types apart from *Monitoring requirements changes*. Another classifier relates positively *Open* triggers with *Human analysis*, which is reasonable considering that in most of the approaches *Open* triggers represent humans making the decision, by any reason, of triggering the adaptation process. From the chronological data shown in Figure 15b and Figure 14b, we have not found relevant information.

A strong positive relation has been found between *Human analysis* and *Human decision* codes (two classifiers relate them; see Table A2-2 in Appendix A2); particularly, in cases when adaptations are executed manually. Regarding the decision-making criteria, a second classifier for *Policies* has resulted, the pattern indicates that in general approaches do not combine *Policies* with *Objective functions* or *Rules* for making decision and that in most of the cases decisions made using *Policies* are execute automatically. Specifically, the relation of *Policies* with *Automatic* enactment makes sense since systems' owner usually design policies for being evaluated and executed automatically and in this way reduce the need of human intervention. Finally, we would like to remark that in general, analysis solutions are developed in an ad-hoc manner while decision-making methods are reutilized by different approaches.

For the types of enactment, we have found a classifier that positively relates *Manual* enactment with *Human decision* criteria, a second one that negatively relates *Human analysis* with *Automatic* enactment, and a third one that indicates that *Semi-automatic* enactment is not usually supported together with the other two types. Given the relations we have found, we can say that approaches in which the adaptation process is started by humans tend to position the whole process in a human-driven manner (i.e., analysis, decision-making and enactment are not automatized). Regarding the chronological information (see Figure 19b), it can be noticed that the involvement of humans in the adaptation process has been retaken after 2009. This is

aligned with the need identified by Cheng et al. [65] and ratified by Krupitzer et al. [2] of considering the users in the adaptation process to ensure their trustiness. However, this participation should be kept as less intrusive as possible so that the automatic adaptation process performance is not affected. Finally, for *Structural* and *Parameter* adaptation, we have found that in general they are not both supported by a single approach, i.e., there is a negative relation between them.

► RQ5. How adaptive monitoring approaches are evaluated?

The distribution of the usage of the different evaluation types over the years is almost the same for all the types. However, *Experimentation* is by far the most used, which is normally the case on scientific papers. Regarding the types of systems in which these evaluations take place, we have found a classifier that indicates that approaches evaluated in *Mobile sensors* are usually found in academic papers and use objective functions as decision-making criterion. Furthermore, those approaches do not tend to trigger adaptations periodically but instead use specific triggers. Particularly, the relation between the type of system and the decision-making criterion makes sense since approaches for *Mobile sensors* in many cases try to solve a trade-off (e.g., distance traveled vs phenomenon understanding) which is usually translated into an objective function. From the chronological perspective, in Figure 22b, it can be noticed that *Sensor networks* popularity has dramatically decreased after 2007 (83,33% of approaches using *Sensor networks* for evaluating their solutions have been published from 2000 to 2007). On the other hand, evaluations in *Clouds/Grids* have suffered the inverse phenomenon (88.88% of approaches have been published after 2007). This can be explained by the emergence of more and more applications for Clouds and Grids in the last years, such the already mentioned IoT.

The last classifier that has resulted from the Data Mining analysis (see Table A2-2 in Appendix A2) does not represent any new insight regarding our understanding about how approaches are evaluated. Instead, it confirms that our coding mechanism is correct, i.e., we expected to find a *No evaluation* code for the type of system in all the approaches coded with *No evaluation* code as type of evaluation. From the literature reviews' point of view, this kind of rules could be beneficial during the data extraction process for checking the correctness of codes assignation and reducing the probability of misunderstandings. Although more experiments with different Data Mining techniques should be performed, the results obtained in this work, regarding the application of Data Mining to qualitative analysis codes, are promising. We have been able to extract meaningful insights, find hidden relations, and analyze review results among different RQs. Moreover, Data Mining has demonstrated to be useful for checking the correctness of the review method's implementation.

To sum up, this SMS has pointed out the lack of generalizability and completeness of current approaches for supporting adaptive monitoring, starting with the lack of a definition for the *adaptive monitoring* term. We have determined, that most of the current approaches supporting adaptive monitoring focus on solving specific problems producing problem-specific

solutions. Moreover, most of current approaches do not support the whole life cycle of adaptive monitoring software systems, i.e., from their design to their deployment, as well as their maintenance. As a first step, we have provided a generic definition for *adaptive monitoring*. In order to construct our definition, we have taken into account our findings about how current approaches conduct the adaption of monitoring systems, i.e., the different types of adaptations supported, triggers, types of monitoring systems and SuM, etc. The generic definition of adaptive monitoring contributed by this thesis RQ, is taken into account for designing our architectural solution for supporting adaptive monitoring in SASs (i.e., contribution of RQ3 of this thesis).

III

How to improve

Study on SASs' self-improvement

Modern software systems are expected to operate under uncertain conditions, without interruption. Possible causes of uncertainties include changes in the operational environment, dynamics in the availability of resources, and variations of user goals. The aim of self-adaptation it to let the system monitor itself and based on its goals reconfigure or adjust itself to satisfy the changing conditions, or if necessary degrade gracefully [66]. Examples of modern systems adopting self-adaptation capabilities go from small-scale smartphones and laptops to large-scale systems-of-systems (SoS) like smart vehicles, production facilities, or data centers [16]. Although, many initiatives have emerged for supporting SASs in the last decades, only recently the first attempts to establish suitable software engineering approaches for the provision of self-adaptation have been made [1]. Thus, many research challenges remain open in this field.

The basis of SASs' self-adaptation ability is the support at runtime of the *self-** or *self-management* properties like: *self-configuration* and *self-healing,* useful for instance, in case of failures; *self-optimization* and *self-protection,* utilized for example, in the presence of threats; or *self-improvement,* used for updating SASs' adaptation logic in order to for instance, respond to an adaptation goal change [5], [16]. Given the great variety of application domains where SASs are utilized, self-* properties have been investigated from different perspectives, such as fault-tolerant computing, distributed systems, biological inspired computing, distributed artificial intelligence, robotics, control theory, etc. [65]. However, almost in all the cases, researchers have focused on the adaptation of the target system and no on the adaptation of the adaptation

logic, i.e., the self-improvement property. This contribution aims at uncovering the challenges that currently affect SASs and the implementation of their self-improvement property as well as the state of the art regarding those challenges. In order to achieve this goal, we have conducted two literature reviews following a systematic protocol. Concretely, we have followed the guidelines of Kitchenham & Charters [36] for SLRs. However, in this case we do not aim at developing an exhaustive SLR with all the work available in the literature, but to report relevant contributions. This contribution of the thesis identifies commonalities and differences of existing proposals for supporting SASs' self-improvement as well as uncovers the current research gaps of this topic. The importance of this contribution is twofold: 1) providing an overview of how self-improvement is implemented by current solutions; 2) turning out new research directions. Most of the contributions presented in this chapter were published in: two deliverables of the SUPERSEDE H2020 European project [19], [20] and the SCI-indexed journal E*xpert Systems with Applications* (I.F.2017: 3.768)[7].

3.1 Introduction to self-adaptation

As it has been explained before, in the introduction of this thesis, SASs are systems composed of two main parts: an autonomic manager (AM) (also referred to as adaptation logic) and the managed elements (MEs) (also referred to as managed resources) [2], [4]. The MEs are the components of the system that provide the main functionality and can be adapted, while the AM corresponds to the control unit that manages the MEs' adaptation process [2]. As introduced in Section 1.1, in practice, the AM is implemented through a feedback control loop such the MAPE-K loop [5], [6]. Investigating SASs, typically consist in finding ways to support the process carried out by the AM. Different taxonomies have been developed over the years for characterizing such process [2]. Recently, Krupitzer et al. [2] have conducted and extensive literature review on self-adaptation and proposed a taxonomy based on the results of the review. The taxonomy consists of five dimensions: R*eason*, *Time*, *Technique*, *Level*, and *Adaptation Control* (see Figure 23).

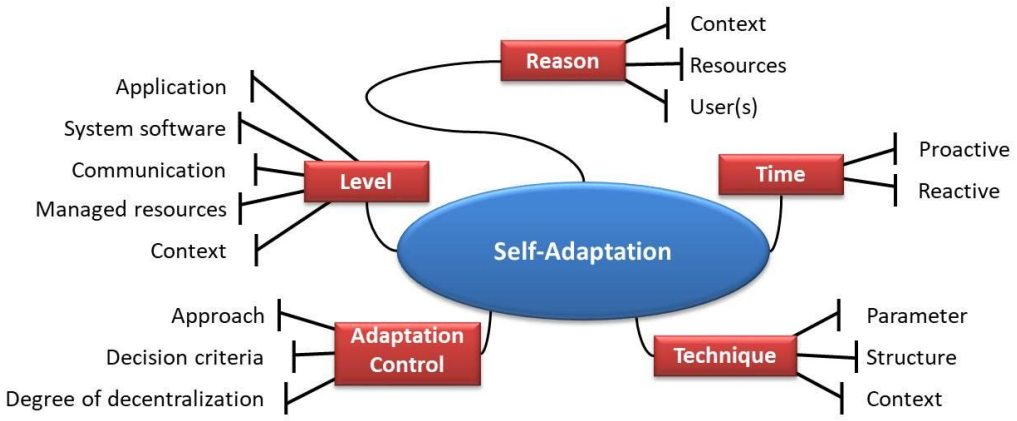


Figure 23: Taxonomy for self-adaptive systems [2]

Below, we describe each of these dimensions:

- *Reason*. The first dimension refers to the reason for an adaptation. For instance, a change in context, in the system's resources, or a change (e.g., changing goals) caused by the user which includes a possible administrator.
- *Time*. The time dimension refers to when an adaptation is executed with respect to a change. This dimension is divided into reactive and proactive.
- *Technique*. This dimension refers to the technique utilized for adapting the MEs. Techniques can be parameter adaptation or structural adaptation (including algorithmic and compositional adaptation). Additionally, in the view of the authors, the context itself can also be adapted.
- *Level*. This dimension refers to the specific level at which an adaptation is executed on a system. As the level of the adaptation, it could be the application itself, the system software, the communication, the technical resources, or the context.
- *Adaptation control*. This last dimension refers to how the adaptation process is controlled in SASs. In this taxonomy, the adaptation control is split into three sub-dimensions: adaptation approach, adaptation decision criteria, and degree of decentralization. The adaptation approach can be internal (i.e., interwoven with the MEs) or external (i.e., separated from the MEs). While, the decision criteria would depend on each approach, some examples are models, rules/policies, goals, or a utility (function). Regarding the degree of decentralization, it may vary depending on whether various subsystems are responsible for controlling the adaptation or the functionality is centralized.

3.2 Open research challenges affecting SASs

Although the extensive efforts that have been spent in different research fields on the realization of SASs, some challenges regarding the capabilities and construction of SASs remain open. In this section, we investigate different works in order to identify these challenges. Particularly, we focus on challenges that affect the operation of SASs at runtime (e.g., challenges about design-time languages for describing SASs will be out of the scope). In the first and second Software Engineering for SASs research roadmap papers [65], [67], the authors identified a set of challenges that affect current SASs and motivate the need for research in different fields (i.e., runtime RE, SASs engineering, runtime verification and validation (V&V), adaptation assurance, etc.). Besides these challenges, Krupitzer et al. [2] added two more important ones regarding the capability of supporting runtime proactive adaptation and context adaptation.

In a more recent work, Weyns [66], speculating on how the field may evolve in the future, presents what he considers the most worth focusing open research challenges. Some of the challenges identified by Weyns, coincide with the ones identified in the works of Cheng et al.[65], De Lemos et al. [1] and Krupitzer et al. [2]. However, others were new, such as the integration of systems' adaptation and evolution processes, and the support of automated

runtime system models. Concretely, from these four resources [1], [2], [65], [66], we have identified the following open research challenges (**Chl**), which we have divided into four categories:

CATEGORY 1 - SASs capabilities challenges

Chl1.1 Provide self-adaptation capabilities to existing systems [65]
Chl1.2 Perform trade-off analysis between several potential conflicting system goals [2], [65]
Chl1.3 Support different adaptation mechanisms (e.g., structural, parameter) for leveraging system capabilities [65]
Chl1.4 Support context adaptation [2]
Chl1.5 Support proactive adaptation [2]
Chl1.6 Integrate system adaptation with evolution for dealing with unanticipated changes [66]

CATEGORY 2 - SASs engineering challenges

Chl2.1 Perform adaptation activities (i.e., monitoring, analysis, decision-making, execution, V&V) without affecting target system performance and availability [2], [65]
Chl2.2 Communicate, coordinate and share AM elements with other SASs [2], [65], [66]
Chl2.3 Support both centralized and decentralized AMs [1], [2], [66]
Chl2.4 Predict the effects of adaptation, e.g., overhead [65]
Chl2.5 Support runtime use of system models (machine-driven) [66]

CATEGORY 3 – SASs requirements challenges

Chl3.1 Capture self-adaptation capabilities and runtime uncertainty in requirements [2], [65], [66]
Chl3.2 Permit requirements and goals monitoring and adaptation at runtime [2], [65], [66]
Chl3.3 Enable dynamic traceability from requirements to implementation [65].
Chl3.4 Consider the user-in-the-loop [2], [65]
Chl3.5 Balance requirements adaptation and assurance such that target system high-level goals are always met [65]

CATEGORY 4 - SASs runtime assurance challenges

Chl4.1 Identify and verify new contexts at runtime for accurately calculating requirements [65].
Chl4.2 Sense and recover from failures [67].
Chl4.3 Integrate V&V activities (e.g., testing, formal verification, adaptation decisions checking, analysis) in the runtime self-adaptation lifecycle (i.e., control loop) [2], [67]

From the challenges listed before, it can be noticed that some engineering and requirements challenges have prevailed over the years, these are: **Chl2.2**, **Chl2.3**, **Chl3.1,** and **Chl3.2**. This does not mean that the rest of challenges have been already addressed by existing approaches

but we believe that these prevalent challenges have particular importance and they must be on the top of the SASs' research community agenda. The requirements challenges, particularly **Chl3.2**, are highly related to the self-improvement property of SASs, while, the engineering challenges are more related to the support of self-adaptation in modern SASs; where decentralization and cooperation are very important factors.

3.3 State of the art on SASs' engineering and requirements challenges

Given their importance, in this section, we analyze, characterize, and compare how existing proposals address challenges **Chl2.2**, **Chl2.3**, **Chl3.1**, and **Chl3.2**. We are concretely interested on investigating whether and how approaches addressing challenges related to self-improvement, i.e., **Chl3.1** and **Chl3.**2, support such functionalities in modern SASs, i.e., **Chl2.2**, **Chl2.3**. In the rest of this section, we provide the details about the protocol we have followed for identifying the proposals. Then, we present our analysis about the state-of-the-art works.

3.3.1. Study identification and selection protocol

The identification and selection process has been guided by the principles of Systematic Literature Reviews (SLRs) defined by Kitchenham and Charters [36]. However, this work does not aim at developing an exhaustive SLR with all the work available in the literature, but to report relevant contributions. Concretely, our selection process has consisted of two main phases:

PHASE 1 – Planning the review

- *Literature resources identification*. In order to identify relevant contributions, we have considered as main literature resources, the studies we have used for identifying the open research challenges in Section 3.2, i.e., [1], [2], [65], [66]. Moreover, we have considered as literature resource, a work [61], highly related to this thesis, presenting state-of-the-art approaches for supporting SASs' requirements adaptation in the presence of uncertainty. This work is related to this thesis as it presents the first ideas that have motivated our proposal. Moreover, the first proof-of-concept implementation of our architectural solution [21], [22], has been based on the ideas of this work.
- *Inclusion/exclusion criteria*. For including and excluding studies, we have designed a series of criteria that we applied to studies' titles, abstracts, as well as full-text reading. Below, we provide the list of the criteria.

  Inclusion criteria

  - Studies present a solution for supporting the adaptation of SASs requirements at runtime
  - Studies present a solution for dealing with runtime uncertainty affecting SASs requirements, through their adaptation

- Studies present a solution for supporting requirements adaptation in decentralized or distributed SASs
- Studies present a solution for adapting SASs requirements and support the communication, sharing, and/or coordination of SASs AM elements with other SASs

Exclusion criteria

- Studies presenting summaries of solutions fulfilling the inclusion criteria, i.e., secondary studies
- Studies not accessible in full-text

PHASE 2 – Conducting the review

- ***Stage 1 - Manual search.*** We have conducted two iterations of manual search. In the first iteration, we have extracted studies from the four literature resources utilized in Section 3.2, i.e., [1], [2], [65], [66]. From the 364 papers (once duplicates were automatically removed) cited by the four resources, 33 articles have been found to be related to challenges Chl2.2, Chl2.3, Chl3.1, and Chl3.2 (see Figure 24). The list of references resulting from this iteration can be found in Appendix B1. In the second iteration, we have included the state-of-the-art approaches presented by Knauss et al. [61], as well as the Knauss et al.'s work itself. The second iteration provided us 10 more papers (see Figure 24). The list of references added in this iteration can be found in Appendix B2.
- ***Stage 2 - Forward snowballing and exclusion by title, abstract and full-text reading***. In this second stage, the applicant has performed a forward snowballing process, in order to identify advances added on top of the approaches found in the manual search, or new emerging approaches. In order to select the final set of papers, the inclusion/exclusion criteria presented in Phase 1 have been applied. Studies have been excluded based on titles, abstracts, as well as full-text reading. Periodic meetings with the supervisors, during the process of selection for discussing the papers, were done.
  In order to decide when to stop the snowballing process, the saturation criterion [49] has been used as follows. When an article did not fulfill the inclusion criteria (or fulfilled the exclusion criteria), the snowballing process, for that article, has stopped, and the article was discarded. When the studies referencing an article that fulfilled the inclusion criteria did not fulfill them (or fulfilled the exclusion criteria), the process for that article has stopped, and the article was added to the final set. From this process, four studies have resulted [61], [68]–[70](see Figure 24). The reference details of these works can be found in Appendix B3.

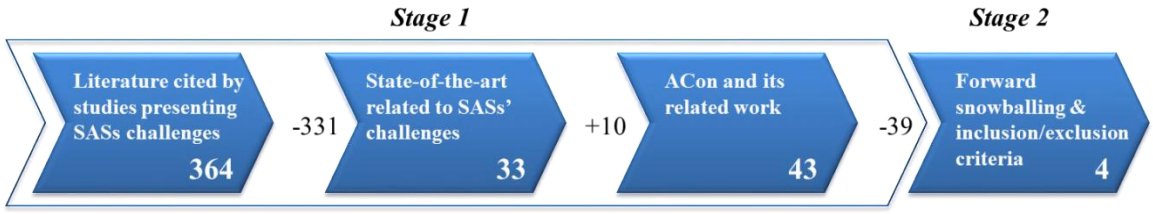


Figure 24: Number of included articles during the selection process

3.3.2. Results of the review

Now, we analyze how the approaches presented in the four works selected [61], [68]–[70], address challenges **Chl2.2**, **Chl2.3**, **Chl3.1** and **Chl3.2**. A summary of the results of this analysis is presented in Table 12.

Table 12
Comparing how approaches address SASs' requirements and engineering challenges

Work by	Chl2.2	Chl2.3	Chl3.1	Chl3.2
Gerostathopoulos et al. [71]	Interaction with other SASs is not supported (not considered)	Decentralization is not supported (not discussed)	Adaptation strategies capture the self-adaptation capabilities	Adaptation strategies monitoring and adaptation at runtime is supported
Klos et al. [69]	Interaction with other SASs is not supported (not considered)	Decentralization is not supported (not discussed)	Adaptation rules capture self-adaptation capabilities. Uncertainty at design time is not captured but managed through runtime models mutation for satisfying goals	Adaptation rules monitoring and adaptation at runtime is supported
Han et al. [70]	Interaction with other SASs is not supported (considered)	Decentralization is not supported (not discussed)	Fuzzy rules capture self-adaptation capabilities and uncertainty.	Fuzzy rules monitoring and adaptation at runtime is supported
Knauss et al. [61]	Interaction with other SASs is not supported (not considered)	Decentralization is not supported (not discussed)	Contextual requirements capture self-adaptation capabilities. Uncertainty at design time is not captured but managed through contextual requirements' re-operationalization at runtime	Contextual requirements monitoring and adaptation at runtime is supported

Gerostathopoulos et al. [71], present a 3-layer architectural solution for increasing the homeostasis of self-adaptive software-intensive cyber-physical systems, i.e., the capacity to maintain an operational state despite runtime uncertainty, by introducing runtime changes to the self-adaptation strategies, i.e., the SASs adaptation capabilities (Chl3.1 and Chl3.2). The architecture is composed of a set of MAPE-K loops that interact between them hierarchically for supporting architectural adaptations of complex SASs and their adaptation capabilities. However, these MAPE-K loops are centralized and details about how they can interact with elements of other SASs' AMs (Chl2.2 and Chl2.3) are not provided.

Klos et al. [69], proposes an extended architecture of the MAPE-K loop for supporting the adaptation of SASs' adaptation capabilities in the form of rules in order to respond to unanticipated environmental changes (Chl3.1 and Chl3.2). The approach utilizes system, environment, and global goal models stored in the Knowledge base (K element of the loop) for automatically evaluate the adequacy of current adaptation rules and delete or generate new rules at runtime through the mutation of runtime models. The MAPE-K loop is extended with two new components: Evaluation and Learning. The decentralization of the elements of the AM and their collaboration with other AMs have not being discussed in this work (Chl2.2 and Chl2.3).

Another interesting work dealing with runtime uncertainty in SASs through the automatic adaptation of SASs adaptation capabilities, is the one presented by Han et al. [70]. This work combines the Analyzer and Planner elements in a single component called Self-learning adapter. Apart from the normal operation, this component is provided with learning abilities that enable it to adapt the rules that capture the SASs' adaptation capabilities, when necessary, in order to handle runtime uncertainty (Chl3.1 and Chl3.2). In order to do that, it analyses runtime sensor data and triggers rules' adaptation based on learnings obtained from that data. The internal approach proposed by this solution could affect the performance of the AM operation, introducing unnecessary overhead. As described by Krupitzer et al. [72], external approaches like the hierarchical adopted by Gerostathopoulos et al. [71], are preferable since they ease the scalability and maintainability of the system. Moreover, this approach lacks of decentralization and collaboration mechanisms, constraining its applicability in modern SASs (Chl2.2 and Chl2.3).

Finally, Knauss et al. [61] proposes the adaptation of SASs' contextual requirements (i.e., adaptation capabilities) for dealing with runtime uncertainty (Chl3.1 and Chl3.2). The proposal relies on a complete and external MAPE-K feedback loop that interacts with SASs for monitoring and adapting their requirements. The advantage of this solution is the possibility of reusing methods and techniques already investigated for SASs' AM, in order to implement the external MAPE-K loop. However, this work does not provide architectural details for constructing such proposal. Moreover, cooperation and decentralization mechanisms are not considered, limiting the demonstration of its value in modern SASs (Chl2.2 and Chl2.3).

3.3.3. Discussion

According to the results of this review, very few approaches have been proposed for improving SASs' operation once they have been deployed; concretely, for automatically monitoring and adapting their requirements (and in consequence their capabilities) as well as dealing with uncertain conditions. Moreover, none of the identified approaches provides the mechanisms required for varying the (de)centralization level of the elements of the AM and supporting the communication and cooperation of AMs of different SASs. As a first step, we provide an abstract idea of how the improvement process of modern SASs could be supported. First, we propose to adopt an external approach in which the improvement process of SASs is managed by other MAPE-K loops (see Figure 25). In this way, existing solutions for MAPE-K loops can be reused. Moreover, adopting an external approach allows systems' owners to manage their SASs and the MAPE-K loops in charge of the improvement process, independently.

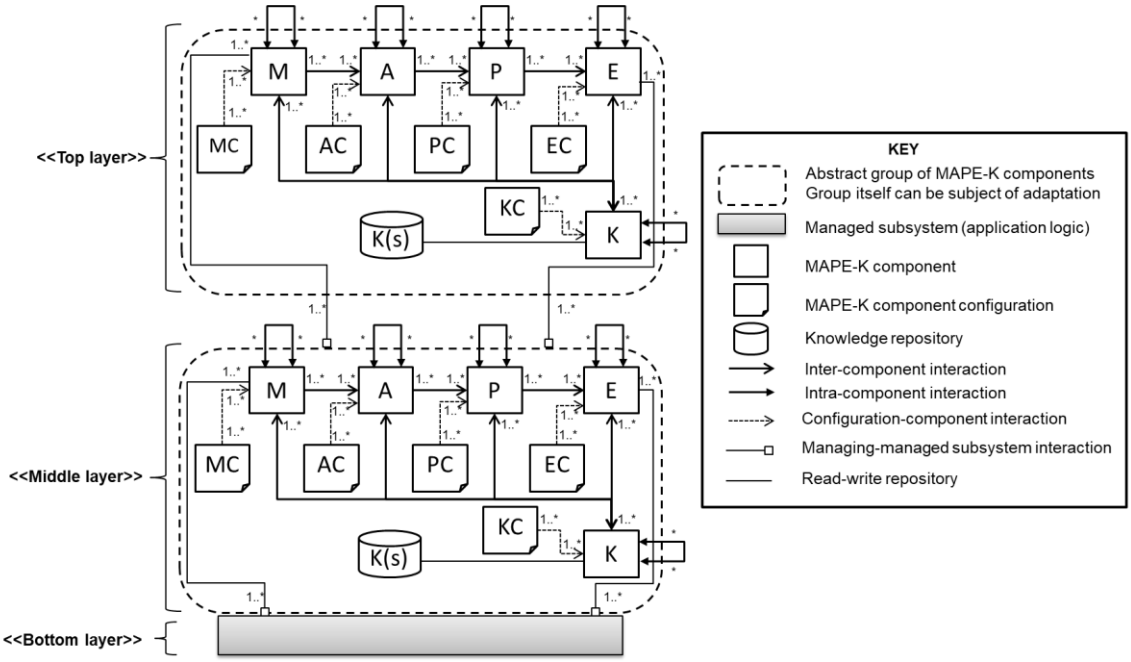


Figure 25: Hierarchical inter-intra collaborative pattern (HIIC)

In order to support the communication between AMs, and different (de)centralization levels, we propose an architectural pattern for designing MAPE-K loops. Concretely, we have extended the notation described by Weyns et al. [4] for decentralized control in SASs and propose a pattern named *Hierarchical inter-intra collaborative pattern* (HIIC) [7] (see Figure 25). This pattern consists of three layers: a bottom layer, corresponding to the MEs; a middle layer, consisting of the AM in charge of the adaptation of the MEs; and a top layer, corresponding to the MAPE-K loop(s) in charge of managing the adaptation of the middle-layer AM operation, in order to better support the MEs. Moreover, this pattern makes explicit the communication of the SASs' AM elements with others of the same or different nature (e.g., a Monitor with other Monitors or

a Monitor with two Analyzers). This characteristic can be seen by observing the cardinalities of the interaction arrows in Figure 25, denominated: *Inter-component interaction* and *Intra-component interaction*. The communication of AM elements with elements of other SASs' AMs, not only allows the cooperation and sharing of elements, but also increases the resilience of modern SASs.

In order to allow systems' owners to vary the (de)centralization level of the elements of the AM, we have incorporated two concepts: *MAPE-K component configuration* and *Configuration-component interaction* (see Figure 25). The configuration elements contain all the knowledge required by each element for performing its functionalities. Apart from enabling the decentralization, the explicit representation of elements' knowledge as a separate element, allows the later adaptation of it and in consequence of the operation of the elements. This architectural contribution has served as a starting point for designing the generic proposal of this thesis for supporting SASs self-improvement (i.e., contribution of RQ3 of this thesis).

3.4 Open research challenges affecting SASs self-improvement

In the SASs research filed, many works have discussed solutions for correctly supporting the adaption process of the MEs [1], [2], [65], [66]. On the contrary, very few works cover the adaptation of the elements of the loop that implement AMs [16], as it has been demonstrated in Section 3.3. According to the analysis presented in Section 3.2 and Section 3.3, enabling the adaptation of these elements would allow modern SASs to address complex challenges such as runtime uncertainty. However, correctly supporting this process entails its own challenges. During the need for a review identification process performed in the literature review that will be presented later in Section 3.5, we have identified the work of Krupitzer et al. [16]. In this work, authors have conducted a comparison of 12 approaches that support SASs' self-improvement. As a comparison metric, authors have utilized the taxonomy for self-adaptation that we have presented before in Section 3.1. The study presents a series of limitations but still we would like to remark one of its outputs, namely a set of open research challenges affecting SASs self-improvement:

Chl1. Future works should elaborate on common, generic strategies to offer more reusable approaches for self-improvement or provide guidelines within use cases and generic guidelines across use cases.

Chl2. Future approaches for self-improvement should consider both reactive (reaction after a change) and proactive (action before a change) adaptation for higher flexibility and improved adaptation results.

Chl3. Future approaches should include structural adaptation of the AM in order to fit better runtime changes, e.g., AM elements' faults.

Chl4. Future works should offer self-improvement in decentralized settings to improve scalability.

Some of these challenges overlap with the challenges identified in Section 3.2 for SASs in general, and with our findings regarding adaptive monitoring systems (i.e., RQ1 of this thesis). In the literature review conducted in Section 3.3, we have focused on approaches that improve SASs through the adaptation of their requirements. In next section, we follow a systematic protocol for identifying existing approaches that support the adaptation of SASs' AM elements, in general. As part of the analysis, we will determine whether and compare how existing proposals address the open research challenges listed above (i.e., **Chl1-Chl4**).

3.5 State of the art on SASs' self-improvement

As we have mentioned before, the literature reviews conducted in this chapter do not aim at developing an exhaustive SLR with all the work available in the literature, as described by Kitchenham and Charters [36], but to report relevant contributions. In this case, we are focusing on quality (sustained by the publishing venues) rather than the quantity of papers.

3.5.1. Need for a review

As stated by Petersen et al. [35], before carrying out any literature study, researchers should identify and evaluate any existing systematic review on the topic of interest. Hence, in order to identify secondary studies on adaptive feedback loops, we have followed a search protocol analogous to the main one presented in the study identification process of our review (see Section 3.5.3). That is, we have searched for existing reviews once the protocol was defined and before the literature review was conducted. In short, we have built a search string as a conjunction of population and intervention, as recommended by Kitchenham and Charters [36], and performed an automatic search on the databases of IEEE Xplore, ACM, Scopus and Inspect/Compendex (Engineering Village).

In software engineering, the population may refer to a specific software engineering role, a category of software engineer, an application area or an industry group [36]. In this review, the population corresponds to studies in the application area of *adaptive feedback loops*. This term can be split into two simpler terms: *adaptive* and *feedback loops*. The intervention is defined as a software methodology, tool, technology or procedure that addresses a specific issue [36]. In our case, the intervention corresponds to a *review*. In order to increase the number of results from each of the terms mentioned before, we have defined a set of synonyms, variants, and acronyms (see Table 13). Due to the amount of possibilities, we have decided to use wildcards for the terms related to adaptation and some of the intervention terms. The search string has been constructed using the resulting terms and the Boolean OR and AND operators, as it is shown in Table 13. The string has been implemented adequately in each database, considering their possibilities and limitations.

The search on the different databases resulted in a final set of 123 papers. Concretely, 39 studies resulted in IEEE Xplore, 21 in ACM, 54 in Scopus and 62 in Inspect/Compendex; after

combining all the 176 studies, 53 duplicates were removed. For the resulting studies, we have applied a study selection protocol similar to the one applied in our review (see Section 3.5.3).

Table 13
Search string

Type	Terms
Population	((adapt* OR self-adapt*) AND ("feedback loop" OR "feedback loops" OR "adaptation logic" OR "adaptation logics" OR "autonomic manager" OR "autonomic managers"))
Intervention	(review* OR survey* OR overview* OR SLR OR "systematic mapping")
Inclusion criteria	Studies present summaries of approaches Supporting the adaptation of AMs in SASs. Studies are in the fields of computer science, systems, or software engineering.
Exclusion criteria	Studies present non-peer reviewed material. Studies are not written in English. Studies are not accessible in full-text.

The only difference is the inclusion/exclusion criteria. Table 13 shows the criteria that we have applied for discarding and including secondary studies. After applying such criteria, only one secondary study related to this review has been identified [16]. The study provides a comprehensive overview of 12 approaches supporting SASs' AM adaptation. However, it presents some important limitations:

- A systematic protocol for the identification and selection of primary studies is not followed. Therefore, relevant works may be missing.
- A rigorous qualitative method for analyzing in-depth primary studies is not presented. Instead, a taxonomy for self-adaptation proposed by the same authors in a previous work [72], is used for characterizing approaches.
- Comparison is performed considering only a single source per approach, except for one approach. Hence, relevant contributions for understanding the research field characteristics, such as maturity and evolution, may be missing.
- A comprehensive understanding about the research field is not provided.

Given these limitations, we consider that conducting a literature review in the adaptive feedback loops topic following a systematic protocol is important and justified. Still, we have decided to reuse one of the outputs of this secondary study [16], namely the set of open research challenges stated at the end of the paper (see Section 3.4). In this review, we analyze whether and how the resulting approaches address such challenges.

3.5.2. Research questions

Given the state-of-the-art, this literature review aims at improving the understanding of adaptive feedback loops in SASs' field. In order to reach this goal, we have designed four main

RQs (see Table 14). These RQs will allow us to perform a deeper analysis, not only of the approaches but also of the research field.

Table 14
Research questions of this review

Research question
RQ1. What are the demographic characteristics of the existing approaches?
RQ1.1. How are approaches distributed over time?
RQ1.2. How are approaches distributed between industry and academy?
RQ2. How is feedback loops adaptation in SASs supported by existing approaches?
RQ2.1. What triggers the SASs' feedback loops adaptation process?
RQ2.2. When is the need for adaptation detected?
RQ2.3. Which technique is utilized for executing SASs' feedback loops adaptation?
RQ2.4. How is the adaptation process controlled?
RQ2.4.1. Which adaptation approach is utilized?
RQ2.4.2. Which criteria are considered for making adaptation decisions?
RQ2.4.3. How are components in charge of the adaptation process organized?
RQ3. In which types of systems can the approaches be adopted?
RQ4. How do current open research challenges relate to the approaches characteristics?

First, in order to improve the understanding about the research field, we have included a demographic RQ that describes the distribution of the approaches: over the years and between industry and academy communities (RQ1). Next, we aimed at characterizing the existing approaches (RQ2). With this purpose, we have used the self-adaptation taxonomy described in Section 3.1. We use the taxonomy dimensions to split RQ2 (see Table 14). Please note that in this review, the *Level* dimension will always correspond to the Application (i.e., the feedback loop implementing the SAS's AM); for this reason, we have not derived any research sub-question from that dimension. In addition, RQ2.4 has been further split into three sub-questions, which correspond to the values of the *Adaptation control* dimension. Finally, in order to understand better whether and how existing approaches address open research challenges cited in Section 3.4, we have designed two more RQs: RQ3 and RQ4.

3.5.3. Study identification and selection protocol

As it has been mentioned before, the identification and selection process of the primary studies has been guided by very well-known principles of SLRs [36] and SMSs [35]. Concretely, our study selection process has consisted of two main phases: *Planning the review* and *Conducting the review*.

PHASE 1 – Planning the review

- *Search string construction*. As in the initial search (see Section 3.5.1), the population of our search is composed by studies in the application area of *adaptive feedback loops*. On the other hand, the intervention in this case is not needed. Therefore, the construction of the search string of the review has consisted on the population terms listed in Table 13.
- *Literature resources identification*. In order to identify relevant contributions, we have searched the top ranked journals in Computer Science, particularly, in Software engineering, Artificial intelligence, Automation and Control systems, and Information systems, based on their JCR Impact Factor (only Q1). We have also included top ranked conferences based on the CORE index (CORE-A and CORE-B). We have considered CORE-B venues since most of the conferences specialized in software adaptation are CORE-B. We have identified 105 journals and 51 conferences. From this set, we have discarded 54 journals and 25 conferences that were not related to our research topic, based on their title and description. As a result, the final set of selected literature resources has consisted of 51 journals and 26 conferences. The complete lists of resources can be found in Appendix B4.
- *Inclusion/exclusion criteria*. The criteria utilized for selecting the primary studies is shown below:

  Inclusion criteria

  - Studies present a solution for supporting the adaptation of AMs in SASs.
  - Studies are published in the last decade (2008-2018).

  Exclusion criteria

  - Studies present work in progress.
  - Studies are not written in English.
  - Studies are not accessible in full-text.

  As it can be noticed, we have limited our search to the last decade, gathering hence the most updated solutions with respect to modern SASs.

PHASE 2 – Conducting the review

- *Stage 1 - Automatic search*. Using the search string designed in the Planning phase, in this stage, we have conducted an automatic search on the selected literature resources. As a result, we have obtained 602 journal papers and 228 conference papers. In total, 830 studies have resulted from this stage.
- *Stage 2 - Exclusion by title, abstract and full-text reading*. From the 830 papers identified in previous stage, we have discarded 817 by title, abstract and full-text reading. The 13 resulting papers conform 11 different approaches: DYNAMICO, ESOs,

RINGA, Generation & evolution of adaptation rules, ACon, SACRE, ActivFORMS, FESAS, Adaptive KBs, Service ensembles and Auto-adjust.

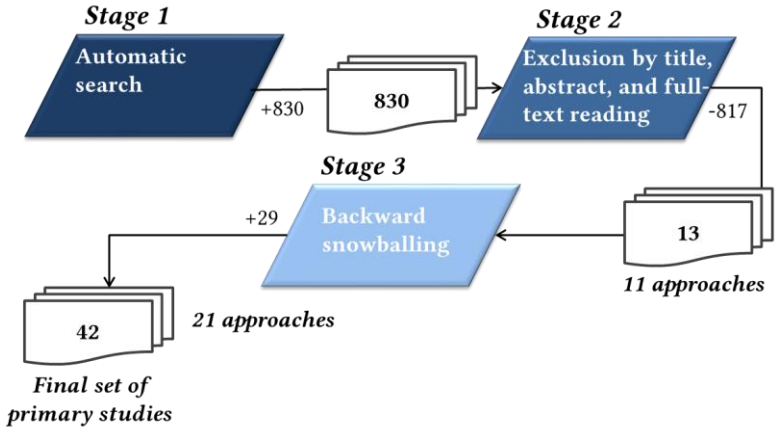


Figure 26: Number of included articles during the selection process

– *Stage 3 - Backward snowballing*. In order to improve the quality of our analysis, we have conducted a backward snowballing process [49]. First, we have considered the secondary study introduced in Section 3.5.1 (also present in the results of Stage 1) and extracted, from its reference list, the works related to approaches that have not appeared in our automatic search. For the sake of simplicity, the secondary study is not included in Figure 26. As a result, 11 new papers have been added to our set, conforming 10 additional approaches: 3LA, NoMPRoL, DCL, PLASMA, FUSION, KAMI, OTC, OTC DPSS, DSPLs and Reqs@RT. Then, on the resulting 24 papers, two more snowballing iterations have been performed for identifying all the studies related to the total 21 approaches. In this process, the publication date has not been restricted, since we were interested on gathering all the studies related. As a result, 18 studies were added to reach a final set of 42 primary studies. The complete list of references can be found in Appendix B5.

3.5.4. Data extraction and visualization

In order to address the RQs of this review (see Table 14), the first author has extracted relevant data from studies following a structured approach based on Miles et al.'s [37] method. A series of meetings have been carried out with the rest of authors for reviewing the resulting data. The reference tool Mendeley and the analysis tool Atlas.ti® have been used for supporting the whole process, ensuring process' consistency, and accuracy. The template utilized for extracting studies' data is shown in Table 15.

Concretely, the data extraction and analysis approach has consisted in two steps: preparation and first cycle coding [37]. In the first step, the primary studies have been imported into a new

Atlas.ti® project. In the second step, primary studies have been coded using the features offered by the analysis tool.

Table 15
Data extracted from primary studies

Data item	RQs	Values
Full reference	-	-
Year of publication	RQ1	-
Type of publication	RQ1	Industry, Academy
Adaptation trigger (Reason)	RQ2.1	Context, Users, MEs (called Resources in [16])
Adaptation time	RQ2.2, RQ4	Proactive, Reactive
Adaptation execution technique	RQ2.3, RQ4	Parameter, Structure, Context
Adaptation control approach	RQ2.4.1	Internal, External
Adaptation decision criteria	RQ2.4.2	Model, Rules, Goal, Utility function
Type of adaptation control elements' structure	RQ2.4.3, RQ4	Centralized, Decentralized
Type of software systems	RQ3, RQ4	-

In this review, some of the codes have been derived from the RQs, i.e., we have performed deductive coding [37], while others, concretely for the data items: full reference, year of publication and types of software system, have progressively emerged from studies during the data extraction process, i.e., we have performed inductive coding [37]. Finally, regarding results visualization, we present our findings in two ways: (1) using figures and tables (see Section 3.5.6), (2) formulating an software engineering theory [73] (see Section 3.5.7). All data extracted from studies is available online at https://goo.gl/oSRXWw.

3.5.5. Validity threats

► Generalizability validity

This study reviews works from different application domains of SASs, from product lines to smart vehicles; therefore, we consider internal generalizability not a major threat. Regarding external generalizability, the results of this review are within the scope of SASs' adaptive feedback loops and we do not attempt to generalize such results beyond that scope. Therefore, this validity threat does not apply to our review.

► Interpretive validity

This validity refers to how reasonable conclusions are given the resulting data. In order to reduce this threat, the insights obtained by the first author as well as possible misunderstandings have been discussed by the other two experienced authors. A series of periodic meetings have been conducted for this purpose.

► Descriptive validity

Descriptive validity is the extent to which observations are described accurately and objectively. For reducing this threat, we have: (1) designed a data extraction template that objectifies the data extraction process (see Table 15), (2) utilized a rigorous qualitative method [37] for extracting the template items from the studies.

► Theoretical validity

Theoretical validity is determined by our ability of being able to capture what we intend to capture. In order to reduce this threat, we have complemented the automatic search with backward snowballing [49]. Our study has not exhaustively reviewed all the work available in the literature; instead, it reports relevant contributions focusing on quality rather than the quantity of papers. Thus, some works may be missing. In spite of this limitation, we consider our sample of primary studies a good representation since 77 high-rated venues, related to the research topic have been considered for performing the automatic search.

► Repeatability validity

In order to ensure repeatability, we have reported the process followed for conducting this review, as recommended by Kitchenham and Charters [36] and Petersen et al. [35]. Moreover, we have used existing well-known guidelines for conducting the review and an existing qualitative method for performing the analysis. Apart from that, we provide a software engineering theory that summarizes and formalizes our results. Therefore, further research can validate, refine, and/or extend our propositions.

3.5.6. Results

In this section, RQs introduced in Table 14 are addressed. Table 16 and Table 17 summarize the answers of the RQs. We have also included the solution that we present in this thesis, named HAFLoop. Before answering the RQs, we briefly describe the 21 approaches identified in Section 3.5.3.

- DYNAMICO [11], [74] relies on three loops in charge of: 1) governing changes in ME's requirements and adaptation properties, 2) preserving ME's adaptation properties, 3) managing the monitoring strategy adaptation for responding to requirements changes, respectively.
- ESOs (Exact-State Observers) [14], [75]–[77] also supports the adaptation of the monitoring strategy as well as the adaptation of the AM enactment process. In order to do that, it relies on a set of pre-defined policies.

- **RINGA** [78] utilizes finite state machines for controlling the adaptation of MEs. If the state machine can no longer respond to environmental changes, a request for re-designing the model is triggered.
- **Generation & evolution of adaptation rules** [79] consists of a reinforcement learning-based framework for: 1) learning adaptation rules offline from different goal settings; 2) evolving adaptation rules online from real-time information about the environment and user goals.
- **ACon (Adaptation of Contextual requirements)** [61] also proposes to utilize learning techniques for adapting SASs' adaptation rules, at runtime.
- **SACRE (Smart Adaptation through Contextual REquirements)** [7], [22] extends ACon with an architectural proposal for supporting the engineering process (from design to implementation) of the approach as well as enabling SASs' AM adaptation in decentralized settings.
- The **Three Layer Architecture (3LA)** [80] is the basis of the layer-based approaches for supporting self-improvement. The proposal consists of detecting situations that cannot be handled by the current system's setup, propagating this information through the different layers, and creating new adaptation strategies.
- The **ActivFORMS** approach [13], [81] follows the architecture proposed by 3LA for managing the adaptation of SASs' formal models, when they cannot deal with the state of the system.
- The **NoMPRoL** approach [82]–[85] also relies on a 3-layer architecture, and probabilistic rule learning for adapting system's model at runtime.
- **Dynamic Control Loops (DCL)** [86] considers SASs with various feedback loops in charge of adapting different parts of the system. Then, it proposes a solution consisting of a framework for dynamically adding and removing loops; and, a modeling technique for designing that kind of systems.
- **PLASMA** [87], [88] proposes a solution for plan-based SASs in which plans are generated/adapted when high-level goals changes or component failures are experienced. It also proposes a layer-based solution.
- **FUSION** [89], [90] supports the development of feature-oriented SASs. It relies on a learning cycle for creating a knowledge base about the impact of adaptation decisions and making better decisions in the future.
- In **KAMI** [91], non-functional requirements models are used to reason about adaptations. These models are updated at runtime using a Bayesian estimator, to fit system's evolution.
- In the **Organic Traffic Control (OTC)** [92]–[95] approach, SASs use evolutionary algorithms for the control of road traffic signals. Therefore, new and unforeseen traffic configurations are generated over time.
- OTC is extended by the **OTC DPSS** [96] approach. In OTC DPSS, intersections collaborate and can form dynamic progressive signal systems (DPSSs).

Table 16
Approaches' characterization

Approach	Time	Reason	Technique	Adaptation control			System type
				Appro.	*Decision criteria*	*(De) centralization*	
DYNAMICO [11], [74]	Reactive	User	Structure	External	Goal	Centralized	OD
ESOs [14], [75]–[77]	Reactive	Context	Structure	Internal	Rules	Centralized	CB
RINGA [78]	Reactive	Context	Parameter	External	Model/ Rules	Centralized	MB
Gen. & evol. of adaptation rules [79]	Both	Context/ User	Parameter	External	Utility	Centralized	GO
ACon & SACRE [7], [22], [61]	Both	Context/ MEs/ User	Parameter	External	Rules	(De) centralized	RD
3LA [80]	Reactive	Context/ MEs/ User	not specified	External	Goal	Centralized	RD
ActivFORMS [13], [108]	Reactive	Context/ MEs/User	Parameter	External	Goal	Centralized	GO and CB
NoMPRoL [82]	Reactive	Context	Parameter	External	Model/ Rules	Centralized	GD
DCL [86]	not specified	User	Structure	External	not specified	Centralized	CB
PLASMA [87]	Reactive	MEs/User	Parameter	External	Model/ Goal	Centralized	GO and CB
FUSION [89]	Reactive	Context/ MEs	Parameter	External	Goal/ Utility	Centralized	GO and MB
KAMI [91]	Both	Context	Parameter	External	Model	Centralized	FO and MB
OTC [92]	Proactive	Context	Parameter	External	Utility	Centralized	MB
OTC DPSS [96]	Both	Context	Both	External	Utility	Decentralized	AS
FESAS [15], [97]	Both	Context/ MEs	Both	External	Rules/ Utility	Centralized	AS
DSPLs [98]	Both	Context/ User	Parameter	External	Model/ Utility	Centralized	(Any)
Reqs@RT [102]	Reactive	MEs/ User	Parameter	Internal	Rules	Centralized	FO and MB
Adaptive KBs [69]	Reactive	Context/ MEs	Parameter	External	Goal/ Rules	Centralized	GO
Service ensembles [105], [106]	Reactive	User	Parameter	External	Rules	Centralized	CB
Auto-adjust [107]	Reactive	Context	Parameter	External	Rules	Centralized	(Any)
HAFLoop	**Both**	**Open**	**Both**	**External**	**Open**	**(De) centralized**	**(Any)**

- The **FESAS** [15], [97] approach enables self-improvement by adding an extended MAPE loop (including a Prediction component) for adapting the AM and a proxy for collecting information from the AM.
- **Dynamic Software Product Lines (DSPL)** [98]–[101] proposes to utilize reinforcement learning for finding new adaptation rules in the configuration space. If the learning is not successful, developers can re-define the DSPL and learning is triggered again.
- **Reqs@RT** [102]–[104] proposes to support SASs' requirements changes at runtime. In this approach, a goal model and an implementation model are maintained. A mapping between these two models allows the correct handling of requirement changes at runtime.
- The **Adaptive KBs** [69] approach has been also identified in the literature review presented in Section 3.3 (Klos et al. approach) where we describe it. In short, it extends the MAPE-K loop by an evaluation and a learning component. Adaptation rules, stored in the Knowledge base, are evaluated at runtime taking into account system goals, and adapted accordingly (removed or added).
- **Service ensembles** [105], [106] utilize two MAPE-K loops. One loop manages the reconfiguration of a service-based system based on requirements (e.g., QoS or required capabilities) while the second one is in charge of adapting such requirements at runtime. In order to do that, the second loop monitors the interaction of users and services.
- The **Auto-adjust** [107] description approach adds a component called Auto-adjust to the MAPE reference model. This extra component continually assesses the current situation using monitoring data and adapts the MAPE elements accordingly. In this approach, all the adaptations consist on adjusting parameters, e.g., the frequency of the loop or the active algorithms.

▶ RQ1 - What are the demographic characteristics of the existing approaches?

- *RQ1.1. How are approaches distributed over time?* The 21 approaches found by our protocol are presented in 42 papers published from 1999 to 2018. All except one of the primary studies (97,4%) have been published from 2006 to 2018. In this period, there has been at least one publication about the topic every year. See Figure 27 for details.
- *RQ1.2. How are approaches distributed between industry and academy?* In order to address this research question, we have extracted the affiliations of all the authors of the 42 studies. The results show that 100% of the authors belong to academic institutions.

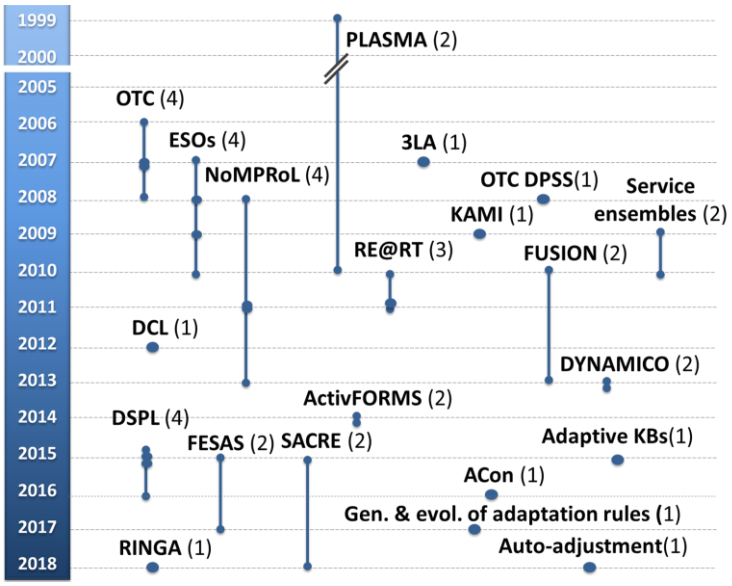


Figure 27: Approaches distribution over time (RQ1.1). Numbers in parenthesis correspond to the number of publications per approach while the circles to their publication year

▶ RQ2 - *How is feedback loops adaptation in SASs supported by existing approaches?*

– *RQ2.1. What triggers the SASs' feedback loops adaptation process?* Regarding what triggers the adaptation of loops, Context, e.g., SAS position, is the most popular trigger (16 approaches, 76,2%), followed by User, e.g., a smart vehicle driver mood (12 approaches, 57,1%), and changes in the MEs, e.g., changes on SAS's topology (8 approaches, 38,1%). Only 4 approaches (19,0%) consider the three triggers. While User and MEs are specific triggers, Context comprises a broader space; therefore, in this work, we have decided to investigate further this trigger. As a result, we have determined that most of the context-driven approaches do not constrain their solutions to a specific variable; instead, they provide examples of them. Figure 28 shows the context variables found and their relevance.

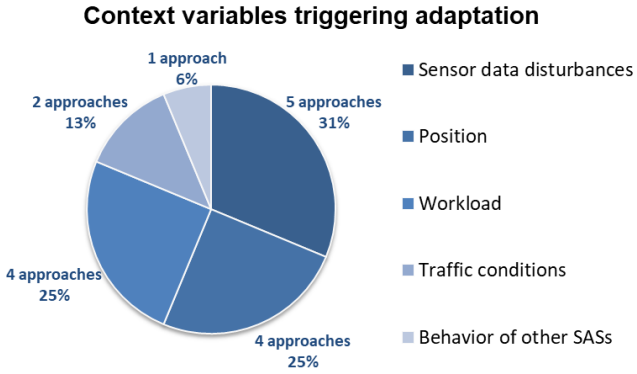


Figure 28: Context variables mentioned by existing approaches (RQ2.1)

- *RQ2.2. When is the need for adaptation detected?* The need for adaptation can be detected proactively, e.g., predicting a lack of resources before it happens, or reactively, e.g., responding to a component fault. Reactive adaptation is the most popular type of adaptation time (19 approaches, 90,5%) while Proactive adaptation is less applied (8 approaches, 38,1%). Most of the approaches supporting Proactive adaptation also support Reactive adaptation (7 out of the 8 approaches, 87,5%).
- *RQ2.3. Which technique is utilized for executing SASs' feedback loops adaptation?* The adaptation technique can be: Parameter, e.g., by changing the frequency of MAPE-K loops' iterations, or Structure, e.g., by deactivating a software component. Parameter adaptation is the most utilized technique (17 approaches, 81,0%) while Structure changes are less supported (5 approaches, 23,8%). Only very few approaches support both techniques (2 approaches, 9,5%).
- *RQ2.4. How is the adaptation process controlled?*
  - *RQ2.4.1. Which adaptation approach is utilized?* The system in charge of the adaptation can be implemented internally, e.g., reutilizing existing MAPE-K loop components, or externally, e.g., introducing new components for exclusively managing the process. External solutions are the most common (19 approaches, 90,5%) while less popular are the Internal (2 approaches, 9,5%).
  - *RQ2.4.2. Which criteria are considered for making adaptation decisions?* In order to conduct the decision making process, most of the approaches utilize Rules, e.g., ECA rules (10 approaches, 47,6%), followed by Utility functions, e.g., a trade-off of adaptation cost and adaptation quality, and Goals (6 approaches, 28,6%, each). The less use criterion is Model (4 approaches, 19,0%). Some approaches consider more than one criterion (6 approaches, 28,6%).
  - *RQ2.4.3. How are components in charge of the adaptation process organized?* The system in charge of the adaptation can manage the process in a Centralized or a Decentralized way (or hybrid). Most of the current approaches, consider Centralized control (19 approaches, 90,5%) while very few incorporate Decentralized settings (2 approaches, 9,5%).

► RQ3 - In which types of systems can the approaches be adopted?

The types of systems found belong to 7 main groups: Goal-oriented (GO) (7 approaches, 33,3%); Model-based (MB) (6 approaches, 28,6%); Component-based (CB) (5 approaches, 23,8%); Requirements-driven (RD) and, particularly, contextual requirements (2 approaches, 9,5%); Feature-oriented (FO) (2 approaches 9,5%); Application-specific (AS), concretely, traffic-control systems (2 approaches 9,5%); and Objective-driven (OD), e.g., SLAs (1 approach, 4,8%). While some of the systems' categories constrain the type of system architecture (e.g., CB), others require the adoption of a specific technique for operating correctly (e.g., GO). Some approaches belong to 2 groups (6 approaches, 28,6%). Finally, we have

identified 2 approaches (9,5%), FESAS and Auto-adjustment, that could be adopted by (almost) any type of SAS.

► RQ4 - How do current open research challenges relate to the approaches characteristics?

In order to determine whether and how the reviewed approaches address the open research challenges introduced in Section 3.4, we have first determined how the items analyzed in previous RQs relate to them. Below, we discuss our findings, organized by challenges. Table 17 summarizes which challenges are addressed by each approach.

- *Chl1*. In order to address this challenge, approaches should be adoptable by a variety of SASs. Therefore, this challenge relates to RQ3's answers. Generic solutions may provide the correct abstraction level to allow other researchers to integrate their problem-specific solutions to their proposals [16]. In this review, most of the approaches do not provide generic solutions; instead, they focus on a specific application domain or specific system type. The FESAS approach proposes the use of MAPE-K loops (with prediction capabilities) for adapting existing ones. The idea is generic enough to be adopted by any SAS. On the other hand, the Auto-adjustment approach proposes a generic component, called auto-adjust, that is added on top of MAPE-K loops for managing the Parameter adaptation of their elements. Regarding reusability, none of the approaches provides support. Moreover, low-level details for implementing the solutions are not provided.
- *Chl2*. This challenge refers to the time at which the need for adaptation is identified and triggered, i.e., it relates to answers of RQ2.2. Approaches should manage adaptations both proactively and reactively. In this review, we have identified 7 approaches supporting both types of adaptation (33,3%). A common solution is to use learning techniques on top of SASs' feedback loops for proactively improving their understanding about context, users, and MEs.
- *Chl3*. This challenges is about enabling the adaptation of the feedback loops' structure at runtime; therefore, it is related to the answers of RQ2.3. According to Table 16, 5 approaches support Structure changes (23,8%) while only 2 of them support both adaptation types (9,5%).
- *Chl4*. This challenge refers to the decentralization degree of the solution, i.e., it relates to answers of RQ2.4.3. In this regard, we have identified 2 approaches that support decentralized settings (9,5%) while only one of them, SACRE, supports a varying degree of (de)centralization.

Table 17
How approaches address current SASs self-improvement challenges

Approach	*Chl1*	*Chl2*	*Chl3*	*Chl4*
DYNAMICO [11], [74]	-	-	✓	-
ESOs [14], [75]–[77]	-	-	✓	-
RINGA [78]	-	-	-	-
Generation & evolution of adaptation rules [79]	-	✓	-	-
ACon & SACRE [7], [22], [61]	-	✓	-	✓
3LA [80]	-	-	-	-
ActivFORMS [13], [108]	-	-	-	-
NoMPRoL [82]	-	-	-	-
DCL [86]	-	-	✓	-
PLASMA [87]	-	-	-	-
FUSION [89]	-	-	-	-
KAMI [91]	-	✓	-	-
OTC [92]	-	-	-	-
OTC DPSS [96]	-	✓	✓	✓
FESAS [15], [97]	✓	✓	✓	-
DSPLs [98]	-	✓	-	-
Reqs@RT [102]	-	-	-	-
Adaptive KBs [69]	-	-	-	-
Service ensembles [105], [106]	-	-	-	-
Auto-adjust [107]	✓	-	-	-
HAFLoop	✓	✓	✓	✓

3.5.7. Discussion

In this section, we discuss the results of the review. First, we present our findings for RQ1. Then, we discuss our findings for RQ2-RQ4 through the formulation of a software engineering theory.

▶ Adaptive feedback loops in SASs research field

According to our findings for RQ1.1, the adaptation of SASs' feedback loops is a lively research area that has gained particular attention in the last decade (36 primary studies, 85,7 %, were published from 2008 to 2018). However, the research field is still not as mature as other related SE areas such the SASs field in which hundreds of contributions can be found [65]–[67], [72]. This can be due to the topic's novelty since it has emerged as a response to the challenges affecting existing SASs and that could have not been addressed with traditional solutions such static feedback loops. The immaturity of the research field is also reflected on the lack of contributions from industry (RQ1.2) as well as the lack of standardized terms for referring to this adaptation process. For instance, some works utilize terms related to the concept of evolution, others use terms related to the concept of learning while others relate this process

with the concept of improvement. This situation could hinder the advancements on the field since existing works are not easily visible to each other.

Given this lack of standardization, in this review we have had trouble for systematically identifying existing work with our previous knowledge (reflected on the search string utilized). Many of the primary studies of this review emerged during the snowballing processes. As a first step towards the creation of a common understanding of this topic, Krupitzer et al. [16] have introduced the *self-improvement* term. However, this new term still has to be adopted by the research community. As an extra step of the field analysis, we have investigated on data trends over time. The results show an increased support of Proactive adaptation, in the last years. Moreover, Context and its combination with the other Triggers have also gained popularity lately. Details about this analysis can be found online at https://goo.gl/YNuYck.

► Theory model

In order to discuss RQ2-RQ4, we have summarized and formalized our results in a theory, following the methodology proposed by Sjøberg et al. [73]. This methodology consists of 5 steps, but for the sake of brevity, in this work, we present the results in a more condensed way. According to Sjøberg et al., a software engineering theory is built by first stating its main constructs (of the archetypes Actor, Technology, Activity and Software System); and then asserting propositions as relationships among constructs. Explanations should justify propositions. Table 18 summarizes the constructs and the explanations of the propositions of our theory while Figure 29 presents the theory, following the UML-inspired notation proposed by Sjøberg et al. [73]:

- Software system refers to the scope of the study. In this review, we are focusing on SASs, particularly, the adaptation of their AMs.
- Technology corresponds to the focus of the review. In our case, we are evaluating adaptive feedback loops. We include different factors on the technology that may influence a particular proposal.
- Activity refers to what the Technology does on the Software System. In our case, it adapts its AM.
- Actor refers to the entity that performs the Activity. In this work, it corresponds to another software system. Our Actor has three relevant attributes, which relate to the open research challenges presented in Section 3.4.

Propositions (P1-P12 in Figure 29) and their explanations (E1-E12) are discussed below, organized by RQ. In order to derive the propositions of our theory, apart from analyzing results of Section 3.5.6, we have investigated trends among solutions using data mining techniques, as we have done in Chapter II. The resulting data mining rules (patterns) can be found online at https://goo.gl/i5jN9e.

Table 18
Constructs and explanations

Constructs		
C1. SAS	C8. System type	C15. Parameter
C2. AM	C9. Context	C16. Centralized
C3. Reason	C10. External	C17. Adapt
C4. Adaptation approach	C11. Generalizability	C18. System in charge of loop adaptation
	C12. Adaptation criteria	
C5. Time	C13. Decentralization	C19. Reusability
C6. Technique	C14. Adaptive feedback loops	C20. Flexibility
C7. Reactive		

Explanations
E1. External approaches have many advantages over internal ones.
E2. Uncertainty is a key challenge of SASs.
E3. To decide the adaptation action, system adaptation capabilities as well as the type of system, must be known.
E4. Designing decentralized systems is more complex than designing centralized systems.
E5. Predicting an event is more complex than reacting to it once it happened, e.g., a component's fault.
E6. To decide the adaptation technique, adaptation capabilities as well as the type of system, must be known.
E7. Flexible solutions should also support decentralized settings.
E8. Re-composing a system, adding or removing components, is more complex than adjusting a parameter.
E9. Generic solutions should be applicable to any type of SAS.
E10. A solution is reusable if it can be applied to a variety of SASs.
E11. Flexible solutions should support both Reactive and Proactive adaptation.
E12. Supporting changes on SASs' structure, apart from parameter changes, improves the flexibility of a solution.

– *RQ2.* According to our findings, many of the approaches reviewed provide External (P1) and Context-aware solutions (P2). Advantages of External approaches are very well-known in the SASs community (E1) [72]. On the order hand, most of the context-aware approaches try to respond to the highly dynamic and uncertain environments to which modern SASs are exposed (E2). In this review, we have presented some of the most common variables: sensor data disturbances, SASs' workload, interaction with other SASs, etc. As mentioned before, uncertainty is one of the most challenging factors affecting SASs [7]. Evidence also indicates that Reactive and Parameter adaptation have been the first and most supported types of adaptation (P5, P8). One can imagine that re-composing a software system at runtime is more complex than simply adjusting a parameter. A typical Parameter adaptation is the adjustment of the adaptation rules or the inference of new ones at runtime (E8). In a similar way, one can imagine that reacting to an event once it happens is easier than trying to correctly predict it, i.e., timely, accurately, etc. (E5).

  Data mining results also show that Proactive and Structure adaptation are typically supported in conjunction with their counterparts, i.e., Reactive and Parameter

adaptation, respectively. The type of SAS may influence the solution to apply; this can be notice more on the Adaptation criteria (P3) and the adaptation Technique (P6). For instance, in a Component-based system, the natural adaptation Technique is Structure (E6) while in a Goal-oriented the decision criteria are the Goals (E3). Finally, the great majority of the approaches propose Centralized settings (P4). In most of the cases, this is done for the sake of simplicity; however, no guidelines are provided for applying the solutions in decentralized systems. This situation is critical since most of modern SASs are deployed in such kind of settings, e.g., IoT systems [65], [72] (E4).

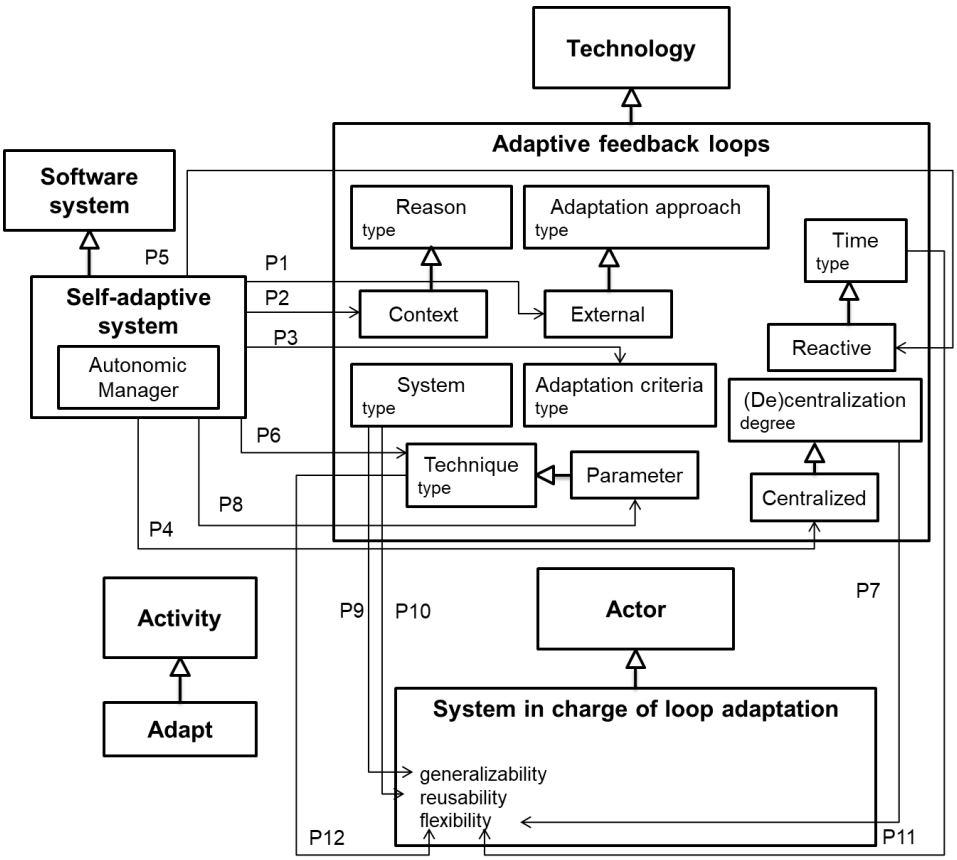


Figure 29: Adaptive feedback loops in SASs' theory model

- ***RQ3.*** The type of software system in which solutions can be applied is directly related to the generalizability (P9) and reusability (P10) aspects of the solution. Most of the approaches have emerged for solving application or domain-specific problems, generating ad-hoc solutions. Therefore, standardized solutions that make the development of adaptive loops for SASs easier and faster are still missing (E9, E10).
- ***RQ4.*** Some characteristics of adaptive feedback loops' solutions are directly related to their generalizability, reusability, and flexibility. As mentioned in RQ3, the System type (P9, P10) is one of them. We can also mention the type of Technique (P12), the degree of Decentralization (P7) and the type of adaptation Time (P11). The relations of these

concepts to the challenges are explicitly stated in Section 3.4 (E12, E7, and E11). None of the reviewed approaches tackles all the open research challenges, being generalizability, reusability, and decentralization the less supported properties. Moreover, the approaches providing these characteristics, do not present a complete proposal for supporting the whole software development lifecycle. Addressing this issue may improve the maturity and visibility of this research field.

This section has presented a literature review on adaptive feedback loops for SASs. The review aimed at providing an overview of this research topic and identifying how existing approaches address current research challenges. In order to achieve this goal, 42 studies organized in 21 approaches, from a variety of domains, were identified, following a rigorous protocol. The primary studies were used for addressing a series of RQs. The results point out the liveliness at the same time as the immaturity of the field. There is a lack of solutions for supporting the development process of modern SASs with adaptive loops. Moreover, the study made evident a gap between researchers and practitioners in this research field. With modern SASs on the rise such as IoT systems, mobile apps, etc., addressing such issues should be a high-priority task to the community members of this research field. The findings of this contribution have motivated the design of HAFLoop, our proposal for supporting modern SASs' self-improvement. In next chapter, we describe our vision of adaptive feedback control loops for SASs as well as our proposal for realizing such vision. The generic solution of HAFLoop is then instantiated for supporting adaptive monitoring in SASs (i.e., contribution of RQ3 and the main research goal of this thesis).

IV

How to support

Building a SAS's self-improvement architecture

The construction of SASs has been studied from different perspectives of the Software Engineering field. Concretely, Weyns [66] has structured these perspectives in six different waves of research: Architecture-based adaptation, Runtime models, Automating tasks, Goal-driven adaptation, Guarantees under uncertainties and Control-based adaptation. The solution that we present in this thesis, HAFLoop, belongs to the Control-based adaptation wave. This wave is concerned with exploiting the basis of feedback control loops for analyzing and guaranteeing key properties of self-adaptive systems, e.g., adjusting SASs' AM at runtime [66]. Control-based adaption is triggered by the complexity to provide assurances (from wave five) and the need for a theoretical framework for self-adaptation (from wave two) [66]. As we have summarized in the reviews presented in Chapter III, there are still gaps to face SASs' AM adaptation in terms of both capabilities and construction.

The aim of this contribution of the thesis is to build a SAS's self-improvement architecture that can fill the gaps pointed out in Chapter III. Concretely, our solution should address the challenges affecting modern SASs' self-improvement process (i.e., challenges **Chl1-Chl4**, identified in Section 3.4). In order to do that, we have first developed an abstract vision about how SASs' self-improvement should be done (Section 4.1). Then, we have specified the key aspects of our architectural solution for materializing such vision. In Section 4.3, we describe the components of our architecture as well as their behavior for supporting adaptive AMs and coordinating such process with other AMs' tasks. In order to do develop our solution, we have taken into account the contributions of RQ2 of this thesis (see Chapter III). The result is a complete, modular, and generic architecture, named HAFLoop, able

to support the designed and development of fully or partially adaptive AMs for SASs. That is, adaptation capabilities are supported for all the MAPE-K elements composing the AM's feedback loop. However, the solutions can be partially adopted for enabling such capabilities to only the components of interest. For instance, a partial proof-of-concept implementation of HAFLoop has been done for supporting the adaptation of SASs' requirements stored in the Knowledge base.

As the main goal of this thesis states, in this research we focus on the adaptation of the Monitor element given the crucial role it plays in the AM. Thus, considering the contributions of RQ1 and RQ2 (see Chapter II and III) we have developed a partial implementation of HAFLoop for supporting adaptive monitoring in modern SASs. This implementation has been evaluated in the domain of the smart vehicles. Different use cases and scenarios for runtime challenging situations such as sensor faults, limited resources, and unpredictable road events were designed and tested. Moreover, experiments were carried out in both simulation and real environments.

Most of the contributions presented in this chapter have been published. The first ideas were published in the applicant's Master thesis [21] and a demo tool session of the *23rd IEEE International Requirements Engineering Conference* (CORE2018: A) [22]. Later, during the development of this thesis, contributions were published in: three deliverables of the SUPERSEDE H2020 European project [23]–[25], a report of the SALI Swedish project (openresearch@astazero program) [26], the tutorials and poster abstracts session of the *BSR winter school - Big Software on the Run: Where Software meets Data* [27], the PhD symposium of the *International Conference on Service-Oriented Computing* (CORE2018: A) [28] and the SCI-indexed journal *Expert Systems with Applications* (I.F.2017: 3.768) [7]

4.1 Our vision of SASs' self-improvement

Following the ideas of our HIIC pattern (see Figure 25 in Chapter III), in order to support the adaptation of the SASs' AM, we consider the MAPE-K feedback loops implementing the AMs as MEs of other MAPE-K loops in charge of their adaptation (see Figure 30). In this way, existing proposals for building SASs could be used (and improved) for adapting the MAPE-K elements. Moreover, in this thesis, we propose to use an external approach. According to previous works in the field of SASs [2], [16], [65], [67], the external design in self-adaption offers several benefits such as:

- ***scalability***, since one MAPE-K loop can manage the adaptation of various MAPE-K loops and vice versa;
- ***maintainability***, as the responsibilities of the loops adapted and the loops in charge of their adaptation are decoupled and can be maintained separately;
- ***reusability***, as the architecture, processes and algorithms can be reused among different loops;
- ***flexibility***, as different degrees of (de)centralization (i.e., decentralized, hybrid, or centralized) are possible.

In a space conformed of AMs and their MEs, systems can be organized at different operation levels, as shown by Figure 30. An AM would be situated in the upper immediate level of which its MEs are placed. In our vision, every MAPE-K loop should be able to: 1) process adaptation instructions for changing its structure or its behavior; 2) manage the adaptation of any type of ME (including MAPE-K loops' elements); 3) collaborate with other feedback loops of the same level for enriching its operation (e.g., improve context knowledge).

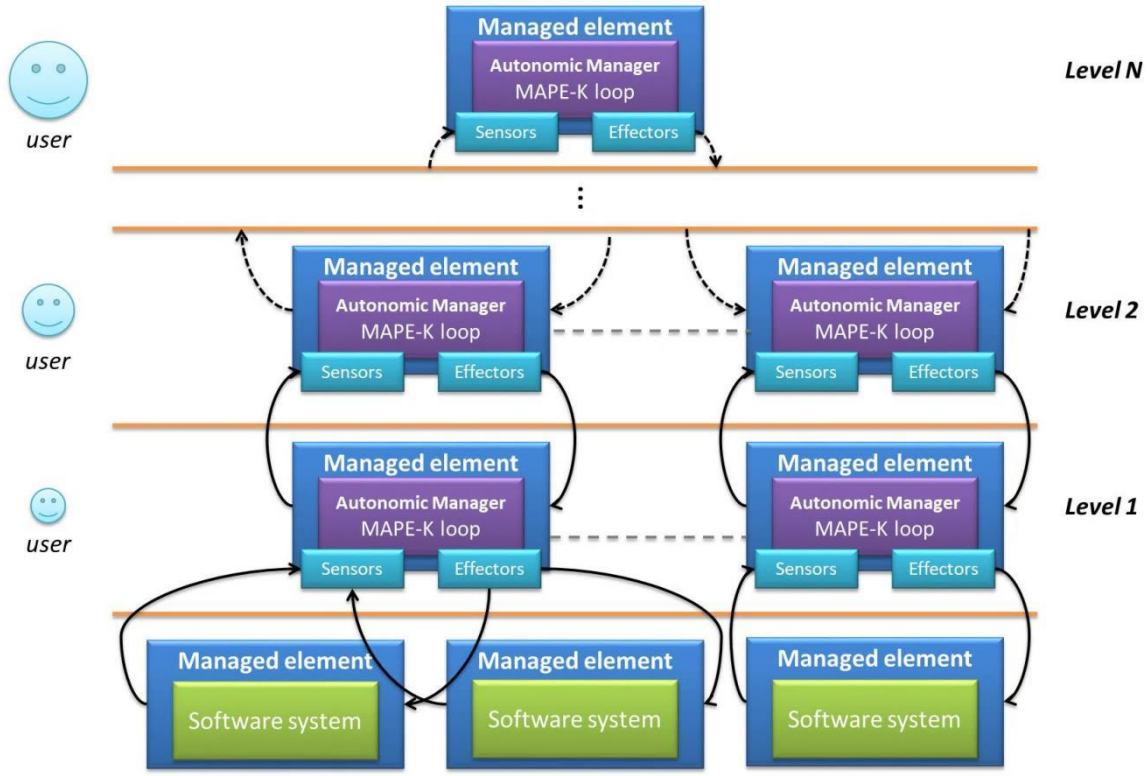


Figure 30: MAPE-K loops adaptation process in HAFLoop

We consider that there could be an unlimited number of levels for the improvement of the adaptation logic of SASs. For instance, a Level-1 MAPE-K loop could utilize simple adaptation rules and policies for driving the adaptation process of the MEs. Then, in a Level-2, a more complex loop with learning capabilities could operate for better supporting the adaptation process conducted by the Level-1 loop. In a Level-3, a loop with more sophisticated techniques could be placed for enriching the operation of the Level-2 loop in the long-term, for instance, using deep-learning techniques. As it can be noticed, levels could be used for growing in complexity and supporting functionalities that require longer iterations. In this way, time-critical adaptation processes performed at lower levels are not affected.

Moreover, with this level-based approach, the adaptation process of a SAS could be gradually enhanced, as levels are added or upgraded over time. Finally, as it is shown in Figure 30, the participation of the user (i.e., system owners, developers, end-users, etc.) at the different levels is considered. Given the description of the levels provided before, we assume that the degree of user involvement will vary from one level to another. At lower levels less user participation may be

required since loops' operation would be simpler and in most of the cases completely automatized; while at upper levels, more complex conflicts may appear, thus higher degree of user participation may be desirable or required.

4.2 A reusable design for MAPE-K loops

Many architecture-based proposals supporting SASs offer reusability on a high abstraction level. However, they neglect reusability on the low component implementation level [109]. Motivated by this fact, Krupitzer et al. propose the FESAS component template [109]. This template separates the generic structure and mechanisms of Autonomic Computing from the custom MAPE elements' functionalities. The contribution is part of the FESAS project framework [110] and aims at simplifying and fastening the development of MAPE elements through reusability of components. The template describes an implementation-independent reusable MAPE element (see Figure 31). As it is shown in Figure 31, a FESAS MAPE element is composed of an exchangeable logic (e.g., for an Analyzer, this would be an algorithm for analyzing monitoring data) and logics for communication and data handling. Moreover, it provides interfaces for receiving and sending data to other components as well as requesting data from other components. The division of communication and data handling functionalities in subcomponents, as well as customized functional logic, enables the reuse of subcomponents among the different MAPE elements as well as different MAPE-K loops.

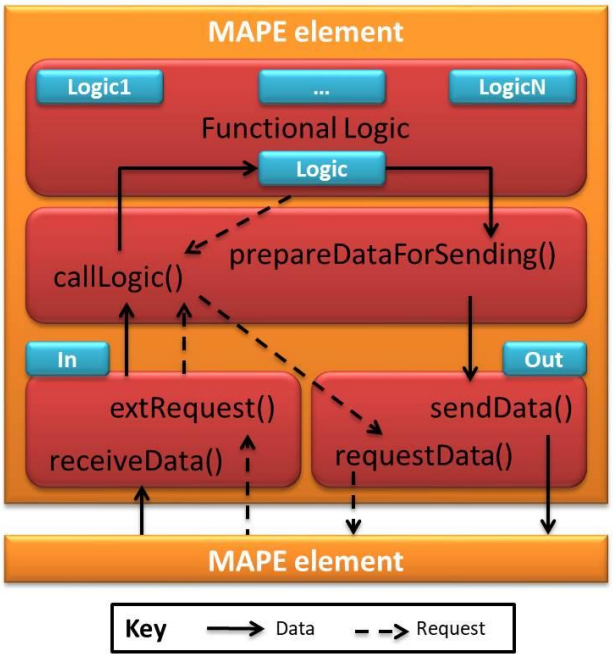


Figure 31: FESAS component template [109]

The FESAS template works as a skeleton of methods for calling the elements' functional logic, communication, and data handling. It is highly reusable since only the functional logic implementation must be customized in each MAPE element. In this thesis, we extend the FESAS

component template with adaptation capabilities for supporting the design and development of reusable adaptive MAPE-K elements. Moreover, we define and provide detailed descriptions of element's components and subcomponents as well as the mechanisms required for coordinating their normal operation with their adaptation process. In our proposal, the Knowledge base element is considered as any other adaptive AM element, thus it is modeled using the same enhanced template. Our design decisions make our proposal a complete and reusable solution for developing adaptive MAPE-K loops in SASs.

4.3 HAFLoop

In this section, we present our architectural proposal called HAFLoop (**H**ighly **A**daptive **F**eedback control **L**oop). Aligned with the principles of the research methodology utilized in this thesis (see Section 1.3), HAFLoop has been developed following an incremental and iterative approach. Nevertheless, we illustrate the results of such development in a linear way for the sake of simplicity.

To design HAFLoop, we have taken into account the open research challenges affecting modern SASs' self-improvement, identified in Section 3.4:

Chl1. Future works should elaborate on common, generic strategies to offer more reusable approaches for self-improvement or provide guidelines within use cases and generic guidelines across use cases.

Chl2. Future approaches for self-improvement should consider both reactive (reaction after a change) and proactive (action before a change) adaptation for higher flexibility and improved adaptation results.

Chl3. Future approaches should include structural adaptation of the AM in order to fit better runtime changes, e.g., AM elements' faults.

Chl4. Future works should offer self-improvement in decentralized settings to improve scalability.

HAFLoop proposes a generic architecture for adaptive MAPE-K loops able to support different adaptation processes (i.e., proactive, reactive, structure, parameter) and system settings (i.e., centralized, hybrid or decentralized), in a variety of SASs. The architecture consists of a set of modular components that can operate together or in isolation. Concretely, we have defined four types of reusable components that correspond to different abstraction levels, from more complex to simpler:

- Adaptive AM or adaptive feedback loop
- Adaptive MAPE-K element
- Element component
- Managers and policies

The term "adaptive" is used in our solution for indicating the ability, but not the obligation, of being adapted. That is, our proposal can be partially implemented for supporting non-adaptive MAPE-K loops or fully implemented for supporting adaptive MAPE-K loops. In this thesis, we emphasize on

adaptive loops; thus, a fully implementation in the form of a framework for Java-based systems, will be presented. Below, we provide more details about each of the HAFLoop components.

► Adaptive feedback loop

Regarding the AM structure, in HAFLoop, we consider that the loop implementing the adaptive AM should have at least one element of each type (i.e., a Monitor, an Analyzer, a Planner, an Executer and a Knowledge base), in order to be able to perform the different tasks and manage the necessary knowledge as to implement adaptation. However, we also consider that it could have more than one element of each type, as we have established in the principles of the HIIC pattern [7]. This could be beneficial in some situations, e.g., for load-balancing, redundancy or for comparing two approaches in a single SAS. AMs could also need to share elements with each other in order to, for instance, coordinate the adaptation decisions of different SASs. This is the case of complex SASs such as traffic control systems, considered SoS [2], [65], [67]. In HAFLoop the communication, sharing and coordination of MAPE-K elements of different AMs is also possible. These structural and behavioral characteristics of adaptive AMs are supported in HAFLoop by the use of runtime policies, configuration elements previously introduced by the HIIC pattern [7]. Policies encompass all the knowledge required by an element for performing its tasks and communicating with other elements of its AM as well as other AMs.

► Adaptive MAPE-K element

In order to address Chl1, we propose a generic architecture for the adaptive MAPE-K elements in which we separated their generic functionality, e.g., communication, adaptation, and data handling tasks, from their specific functionality, i.e., the logic required to monitor, analyze, plan, execute, and manage runtime knowledge. The HAFLoop MAPE-K element architecture extends the FESAS template [109] presented before, with a set of components in order to coordinate MAPE-K elements' normal operation with the execution of adaptation instructions at runtime. Concretely, a HAFLoop MAPE-K element is composed of four functional layers: a *Communication layer*, a *Message processing layer*, a *Logic layer,* and a *Knowledge layer* (see Figure 29). These layers represent the main functionalities of an element, i.e., communicate with other components, process input and output messages (including data and requests), execute element-specific or adaptation logics and manage runtime knowledge, respectively.

The *Communication and Message processing layers* correspond to the communication and data handling components in the FESAS template. While, the *Logic layer* partially corresponds to the logic component, since adaptation capabilities are not considered in the FESAS template. Finally, we have extended the original template with a *Knowledge layer*, in order to enable MAPE-K elements to manage element-specific runtime knowledge such as policies, which also play a crucial role in the elements' adaptation process. In order to perform the functionalities described above, each layer relies on one or more components (see Figure 32). Concretely,

– *Communication layer*. This layer is composed of a Sender and a Receiver component.

- ***Message processing layer***. This layer consists of a Logic selector and a Message composer component.
- ***Logic layer***. This layer is composed of a Functional logic and an Adaptation logic component.
- ***Knowledge layer***. This layer consist of a single component called Knowledge manager.

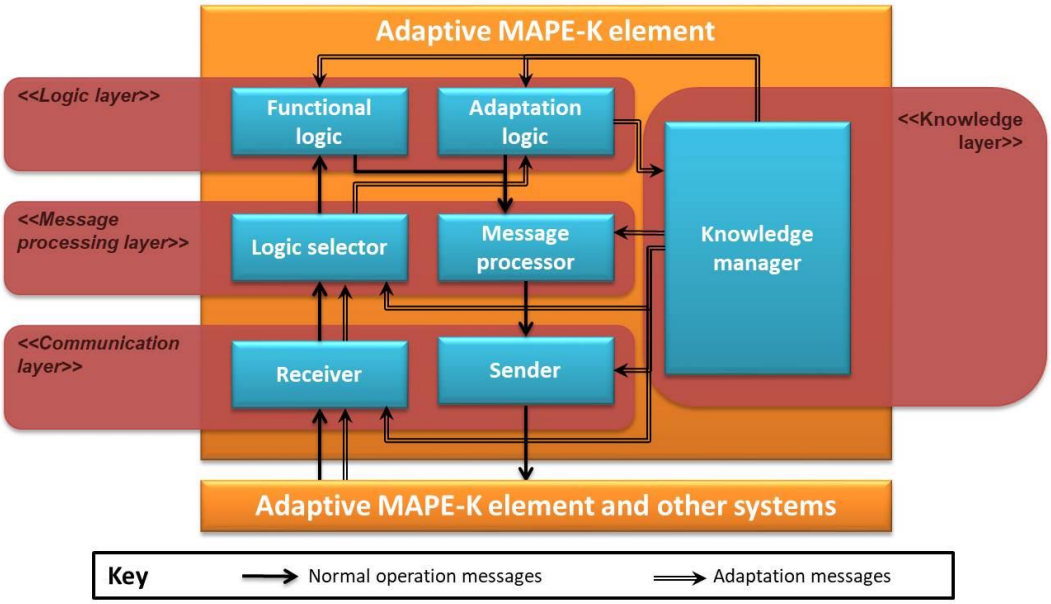


Figure 32: HAFLoop adaptive MAPE-K element

Below, we describe each of these components.

- ***Receiver***. The *Receiver* component is in charge of providing an interface for enabling external systems (e.g., other loop elements or the MEs) to communicate with the elements. It receives all input messages and forwards them to the *Logic selector* component.
- ***Logic selector***. The *Logic selector* component, in its turn, analyzes the input messages and selects the logic component that should process them, i.e., the *Functional* or the *Adaptation logic* component.
- ***Functional logic***. The *Functional logic* is the component in charge of enacting any logic related to the main functionality of the elements and is what gives them their nature, i.e., it determines whether an element is a Monitor, an Analyzer, a Planner, an Executer or a Knowledge base. Output messages produced by this component are sent to the *Message composer* component where they are further processed.
- ***Adaptation logic***. This component contains the logic for processing adaptation request messages received from the *Logic selector*, for instance, it could decide whether an adaptation action can actually be enacted or not given a specific context. The *Adaptation logic* forwards adaptation requests to the *Knowledge manager* for being executed, and, if needed, sends output messages (e.g., an acknowledgement of the received adaptation request) to the *Message composer*.

- ***Message composer***. The Message composer component is in charge of preparing elements output messages. Output messages are mainly generated by the Functional and the Adaptation logic components. Concretely, the Message composer's function consists in determining the recipients of a specific message, ensuring format adequacy, and creating the necessary message copies. These copies are passed to the Sender component for being forwarded to the final recipients.
- ***Sender***. The main function of this component is to send output messages to the corresponding recipients (e.g., other loop elements or the MEs).
- ***Knowledge manager***. The *Knowledge manager*, as its name implies, is in charge of managing the knowledge required by the rest of components for operating correctly. In our proposal, knowledge is stored in the form of runtime policies, which can be adapted at runtime. In order to support the adaptation process, this component receives adaptation requests from the *Adaptation logic*, then it determines to which component(s) the requests should be forwarded. This component can also be utilized for managing other types of knowledge. However, in this thesis we focus only on the adaptive runtime policies since they play a crucial role in the adaption of the MAPE-K elements.

Element component

In HAFLoop, element's adaptations are managed at the component level. This decision makes our design modular and scalable since each element's component can be adapted completely independent from the rest of the element's components. In order to manage both the normal operation and the adaptation process, we propose to include in each element's component, three subcomponents: a *Message manager*, a *Component policy manager,* and a *Component policy* (see Figure 33). The *Message manager* subcomponent is dedicated to receive normal operation messages from other components (or external systems, e.g., in the case of the *Receiver*). The *Component policy manager* receives adaptation messages from the *Knowledge manager*. After processing the adaptation message, the *Component policy manager* sends the corresponding adaptation action to the *Component policy* subcomponent. This last subcomponent represents the current active policy. After receiving the adaptation action, the *Component policy* performs two tasks: first, it updates the component-specific policy variables; second, it notifies the changes to the rest of subcomponents that utilize the policy variables, e.g., the *Message manager*.

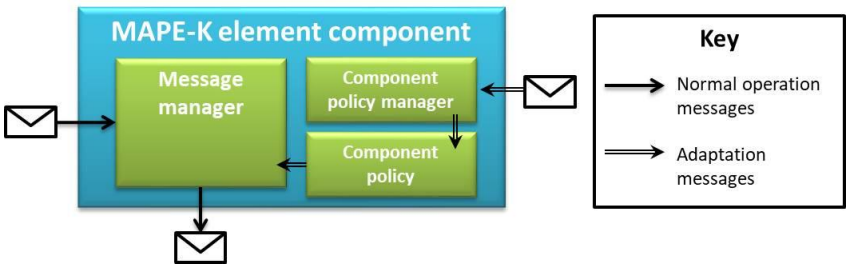


Figure 33: HAFLoop element component

Managers and policies

In order to perform the component-specific tasks described before, each component's *Message manager* should be implemented in a customized way. In Figure 34, we propose a set of subcomponents for implementing the different *Message managers*. As it can be seen, some *Message managers* are more complex. Below, we describe these subcomponents:

- ***Receiver***. The *Message manager* of the Receiver is implemented by the *Message processor*. This subcomponent utilizes the *Receiver*'s policy for deciding to which *Logic selector* a message should be sent and sends the message.
- ***Logic selector***. The *Message manager* of the *Logic selector* is implemented by the *Message dispatcher*. This subcomponent utilizes the *Logic selector*'s policy for deciding to which logic a message should be sent, i.e., the *Functional* or the *Adaptation logic*, and sends the message.
- ***Functional logic***. The *Message manager* of the *Functional logic* is implemented by the *Functional logic enactor*. The enactor's functionality consists in calling the *Functional logic enactor manager* for deciding to which specific logic a message should be sent. The implementation of the *Functional logic enactor manager* as well as the available logics should be done by each HAFLoop instance. Apart from updating its policy variables, the enactor transmits those changes to the manager, which in its turn may decide to which specific logic(s) these changes should be communicated.
- ***Adaptation logic***. The *Message manager* of the *Adaptation logic* is implemented by the *Adaptation logic enactor*. This subcomponent utilizes the *Adaptation logic*'s policy for deciding whether an adaptation can be enacted, given the current context, and to which *Knowledge manager* the accepted adaptations should be communicated. The *Adaptation logic enactor* sends then the corresponding message.
- ***Message composer***. The *Message manager* of the *Message composer* is implemented in part by a *Formatter* subcomponent, which utilizes the *Message composer's* policy for determining: 1) to which recipients a specific message type should be sent, 2) which data format is required by each of the message recipients. Then, a *Message creator* subcomponent, which receives requests from the *Formatter,* generates an output message per each request. Using the *Message composer's* policy, the *Message creator* sends the messages to the corresponding *Sender*.
- ***Sender***. The *Message manager* of the *Sender* is implemented by the *Message sender*. The logic of this subcomponent should be implemented by each HAFLoop instance in order to allow the MAPE-K element to communicate with external systems, i.e., considering the different interfaces required/available for those interactions. It is advisable to use the *Sender*'s policy for conducting this task.
- ***Knowledge manager***. The *Message manager* of the *Knowledge manager* is implemented by the *Adaptive knowledge manager*. This subcomponent utilizes the *Knowledge manager*'s policy for deciding to which element's component an adaptation should be sent, and sends the message.

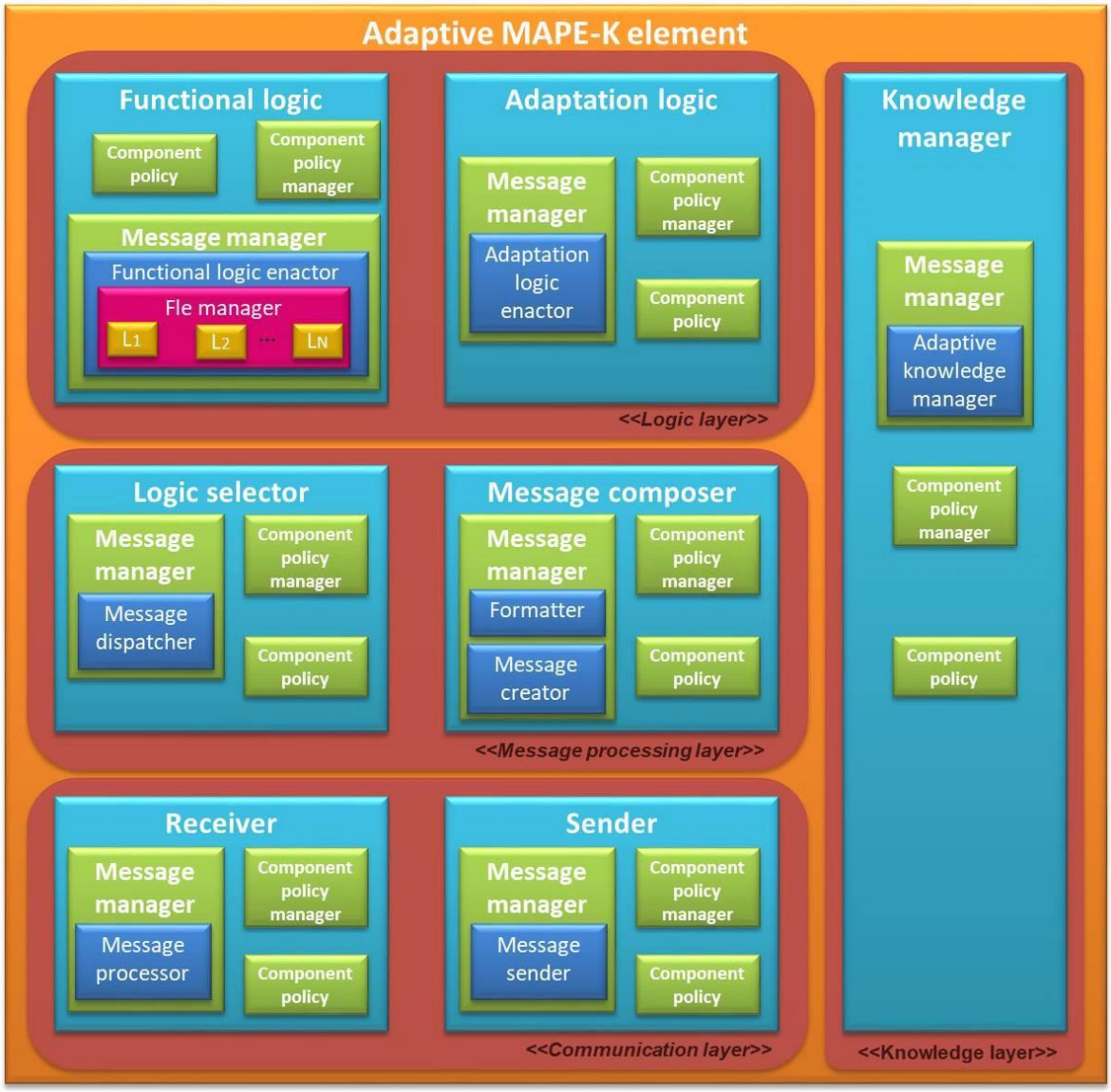


Figure 34: HAFLoop adaptive element components and subcomponents

Extending the HIIC pattern [7] to a lower level, in HAFLoop, similarly to at the AM level, at the element level, more than one type of component (*Receiver, Logic selector, Functional logic, Adaptation logic, Message composer, Sender and Knowledge manager*) can be present. The only consideration is that at least, one component of each type is necessary for setting up an adaptive element. The advantages of allowing different MAPE-K elements' structures are similar: redundancy, load balancing, etc. In this case, the decision of considering component-level policies is what allows systems' owners to design elements with different structures (e.g., an Executer with two Sender components for managing separately messages of two MEs). Moreover, since each component and element has all the knowledge it requires in its policies, different loop, and element settings are possible, from centralized to fully decentralized, addressing Chl4.

HAFLoop components' operation is driven by a set of adaptive runtime policies. The configuration variables contained in policies are intended to describe both how an element, and in consequence its components, should behave and be structured. This allows HAFLoop instances' owners to focus only on how every policy adaptation should be translated into changes of their specific components and not on how to manage the adaptation process. Policies can contain innumerable configuration variables; variables will depend on the requirements of each HAFLoop instance, i.e., each use case. Variables can be generic and reusable among different SASs, but also domain specific. In Table 19, we provide a list of element-independent variables that could be included in policies. While, in Table 20, we extend this list with a set of element-specific variables that could be included in policies. These lists of variables do not intend to be complete, but to serve as guideline for future approaches adopting HAFLoop.

Table 19
Element-independent policies

Element component	Component policy variables
Receiver	– List of Logic selectors to which input messages could be sent – Criteria for deciding to which Logic selector(s) an input message should be sent
Logic selector	– List of Functional and Adaptation logics to which input messages could be sent – Criteria for deciding to which Functional and Adaptation logic(s) an input message should be sent
Functional logic	– List of Messages composers to which output messages could be sent – Criteria for deciding to which Messages composer(s) an output message should be sent
Adaptation logic	– List of Messages composers to which output messages could be sent – List of Knowledge managers to which input messages could be sent – Criteria for deciding to which Messages composer(s) an output message should be sent – Criteria for deciding to which Knowledge manager(s) an input message should be sent
Knowledge manager	– List of Receivers, Logic selectors, Functional and Adaptation logics, Messages composers and Senders to which adaptation messages should be sent – Criteria for deciding to which kind of element's component an adaptation request should be sent and to which specific Receiver(s), Logic selector(s), Functional and Adaptation logic(s), Messages composer(s) or Sender(s) an adaptation request should be sent (in case the adaptation is not for the Knowledge manager itself)
Message composer	– List of Senders to which output messages should be sent – Final recipients and format(s) accepted by each of them. – Criteria for deciding to which Sender(s) an output message should be sent
Sender	– List of external recipients to which output messages should be sent – Criteria for deciding to which external recipient(s) an output message should be sent

Table 20
Element-specific policies

Adaptive element	Element component	Component policy variables
Monitor	Functional logic	– List of data gathering instruments or monitors (sensors, services, logs) – List of variables to be monitored – Monitoring variables characteristics (e.g., type of variable, thresholds, etc.) – Data gathering instruments' mechanism (pull, push) – Monitoring frequency (per monitor or variable) – Monitoring cost
	Sender	– List of Analyzers, Knowledge bases and MEs to which output messages could be sent – Criteria for deciding to which Analyzer(s), Knowledge base(s) or ME(s) an output message should be sent
Analyzer	Functional logic	– List of analysis instruments (tools, algorithms, techniques, etc.) – Analysis' parameters (constraints, evaluation parameters, etc.)
	Sender	– List of Planners and Knowledge bases to which output messages could be sent – Criteria for deciding to which Planner(s) or Knowledge base(s) an output message should be sent
Planner	Functional logic	– List of planning/decision making instruments (tools, algorithms, techniques, etc.) – Planning instruments' parameters (objective functions, evaluation parameters, etc.)
	Sender	– List of Executers and Knowledge bases to which output messages could be sent – Criteria for deciding to which Executer(s) or Knowledge base(s) an output message should be sent
Executer	Functional logic	– List of MEs that could be adapted – MEs' adaptation enactment requirements (e.g., adaptation request format) – Criteria for deciding to which ME(s) and adaptation request should be sent
	Sender	– List of Knowledge bases and MEs (and their Effectors) to which output messages could be sent – Criteria for deciding to which Knowledge base(s) or ME(s) (through its/their Effectors) an output message should be sent
Knowledge base	Functional logic	– List of data stores for persisting runtime data and the format in which data should be persisted (e.g., json, relational tables, etc.) – List of data types to be persisted (e.g., sensor data, analysis alerts, etc.)
	Sender	– List of Monitors, Analyzers, Planners and Executers to which output messages could be sent – Criteria for deciding to which Monitor(s), Analyzer(s), Planner(s) or Executer(s) an output message should be sent

The policy variables listed in Table 19 and Table 20 can be adapted at runtime. According to these lists, a *Sender*'s policy could be adapted for changing the loop structure, e.g., changing the Analyzer to which a Monitor has to send messages, or add a new Knowledge base (addressing **Chl3**). On the other hand, *Functional logic*'s policy could be adapted for changing the behavior of the AM elements, e.g., changing the monitoring frequency, the analysis algorithm or the decision-making evaluation parameters. Policies are also useful for enabling feedback loops to support the addition and removal of MEs at runtime. For instance, when a ME is added, its corresponding policy variables can be communicated to the AM elements using the same mechanism as the one applied for policies' adaptation.

HAFLoop architecture can be utilized for supporting both, proactive and reactive adaption techniques (partially addressing **Chl2**). In order to exemplify how **Chl2** could be address by approaches adopting the HAFLoop architecture for constructing SASs, in this thesis we provide an evaluation on the smart vehicles domain where different scenarios involving proactive, reactive as well as structural and parameter adaptation are tested. Finally, thanks to the external approach adopted by vision, HAFLoop MAPE-K loops in charge of self-improvement operate independently from loops in charge of the MEs' adaptation; therefore, no overhead on MEs' adaptation process is introduced. Instead, considering adaptive feedback loops may help existing SASs to deal with challenging factors such as faults and uncertainty. In this thesis, we focus on the performance of self-improvement loops, as we will describe in next sections dedicated to the evaluation of HAFLoop.

4.4 SACRE: a proof-of-concept

SACRE (**S**mart **A**daptation through Contextual **RE**quirements) [7], [22] is a proposal that we have developed for supporting the detection of modern SASs' contextual requirements (i.e., adaptation rules) affected by uncertainty, and the application of Machine Learning techniques to determine the best operationalization of context based on sensed data, at runtime. This approach is a step forward of the approach ACon [61] which contributed with the ideas of the techniques to be used for supporting SASs under uncertain conditions. SACRE has primarily focused on architectural decisions, its main contributions is the HIIC pattern that we have mentioned in RQ2.

Summing up, SACRE provides the first, and some of the fundamental, ideas of HAFLoop, applied to the field of runtime requirements engineering for SASs. Concretely, SACRE adopts the HIIC pattern as follows. At the bottom layer, MEs are placed; at the middle layer, an AM in charge of the adaptation of the MEs (through the evaluation of contextual requirements) operates; and at the top layer, a second MAPE-K loop is in charge of adapting the contextual requirements utilized by the middle layer AM, when runtime uncertainty is experienced. In order to evaluate SACRE, we have conducted an evaluation using different uncertainty scenarios in real-time in the domain of smart vehicles.

4.4.1 The smart vehicles domain

Smart (or intelligent) vehicles are systems capable of sensing contextual data (e.g., from the driver, the environment, the vehicle itself) and making decisions based on these data (e.g., turn on an alarm or activate self-driving functionality). These systems have become increasingly popular in the automotive industry. In consequence, the interest of the research community in this domain has increased steadily. Smart vehicles have brought several societal benefits, for instance, improving drivers' safety, optimizing fuel consumption, improving driver's experience and comfort. At the same time, their control systems need to face challenging characteristics such as runtime faults, uncertainty, or limited resources, what make this domain still subject of research. In SACRE, we have used contextual requirements for describing a particular functionality of a smart vehicle involving self-adaptation capabilities. This functionality consists of the detection and support of drowsy drivers. A contextual requirement is defined as follows:

> *"A contextual requirement consists of a 2-tuple of the <<expected system behavior>> and the specific <<context>> within which this expected behavior is valid"*
>
> ***Knauss et al.**, Acon: A learning-based approach to deal with uncertainty in contextual requirements at runtime* [61]

▶ Contextual requirements for detecting and supporting drowsy drivers

Many of the road accidents are occurring due to driver fatigue (i.e., driver drowsiness or driver sleepiness). Sleepiness reduces the concentration, activeness, alertness, and vigilance of the driver and it makes the driver to take slow decisions and sometimes no decisions at all. Drowsiness affects the mental alertness and decreases the ability of the driver to operate a vehicle safely, increasing the risk of human error that could lead to fatalities and injuries. Hence, to increase the road safety, there is a need to address this issue to avoid accidents by alerting the driver [111]. In order to do so, the state of drowsiness and alertness of the driver should be effectively monitored [112].

Drivers can experience different levels of drowsiness and alertness, from drowsy to dangerously drowsy and finally sleeping. In order to prevent accidents, different mechanisms for alerting and supporting drivers have been used, for example, in previous a work [113], auditory and seat-based vibration warnings have been proposed to mitigate driver distraction. Other examples are lane keeping assistance systems and lane departure avoidance systems studied by many researchers [113]. In SACRE, in order to detect and support drowsy drivers at different stages, we have defined three levels of drowsiness with three different actuators to support each of these levels. In order to be supported by SACRE, we have modeled them as contextual requirements (see Table 21).

The different drowsiness levels correspond to the <<*context*>> of the requirement, while the activation of the actuators corresponds to the <<*expected system behavior*>>. The satisfaction of the contextual requirements consists of the execution of the corresponding expected system behavior when a context holds at runtime. In order to evaluate contexts continuously, SACRE uses sensor data.

Thus, situations such de-calibration of sensors or faults can cause contextual requirements dissatisfaction. Below, we describe the set of sensors used for the operationalization of the contextual requirements' context listed in Table 21.

Table 21
Contextual requirements

Id	Context	Behavior
cr_1	Driver is drowsy	Activate seat-vibration alarm
cr_2	Driver is dangerously drowsy	Activate sound-light alarm
cr_3	Driver is sleeping	Activate lane keeping support

Sensors

According to a previous work [112], there are three types of measures that have been used widely for monitoring drivers' drowsiness:

- *Vehicle-based measures*. These include deviations from the lane position, movement of the steering wheel, pressure on the acceleration pedal, etc. Once crosses a specified threshold, it indicates a significantly increased probability that the driver is drowsy.
- *Behavioral measures*. For example, yawning, eye closure, eye blinking or head position, usually monitored through a camera.
- *Physiological measures*. Namely, electrocardiogram (ECG), electromyogram (EMG), electrooculogram (EoG), and electroencephalogram (EEG).

For the development of an efficient drowsiness detection system, the strengths of the various measures should be combined into a hybrid system [112]. In the evaluation of SACRE, we use three different (simulated) sensors for obtaining measures from the three types mentioned above. Below, we describe each of the sensors:

- *Steering wheel pressure sensor*. The lack of hands or only one hand on a steering wheel could be an indication of a drowsy driver [114]. Based on the patent presented by Lisseman et al. [114], we consider a steering wheel pressure sensor (triangles in Figure 35) for continuously obtaining the number of driver's hands on the steering wheel (*hosw*).
- *Driver's vigilance level camera*. Eyes closure during long periods and frequently non-frontal face position are clear symptoms of driver fatigue [115]. Camera-based sensors for monitoring drivers in real time, as the prototype presented in a previous work [115], have been proposed for extracting behavioral measures such the ones mentioned before. In SACRE, we have considered a camera sensor (circle in Figure 35) able to report the eyes' state, indicated by the percentage of the driver's pupils that is visible, and the *face position*, with two possible values frontal and non-frontal. Then, using the eyes' state variable we have calculated the percent eye

closure (*perclos*) measure, consisting in the percentage time where driver's pupils have been less than 20% visible.

– ***Electrocardiogram***. As driving, fatigue develops heart rate slows, triggering a series of events (i.e., blood pressures go down, poor circulation and finally hypoxia in brain) that induces drowsiness and loss of concentration [116]. More and more non-intrusive electrocardiogram sensors, as the one presented by Wartzek et al. [117], are developed for being incorporated in vehicles in order to continuously monitor drivers' heart rate. Based on Wartzek et al. prototype, in this work we consider an electrocardiogram sensor (squares in Figure 35) for obtaining the physiological measure hearth beats per minute (*hbpm*).

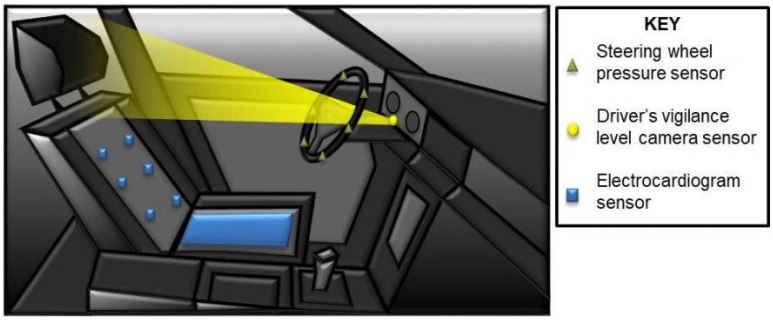


Figure 35: Smart vehicle sensors: steering wheel pressure sensor, camera, and electrocardiogram

The measures obtain by the sensors (i.e., environmental variables) listed above (i.e., *hosw, perclos, face position* and *hbpm*) are combined by expression operators (i.e., relational, arithmetic, and logical) in order to operationalize the contextual requirements' contexts. For example, a context operationalization of cr_1 (i.e., driver is drowsy, see Table 21), could be:

```
perclos >= 15 AND hbpm >= 67 AND hbpm <= 72
```

Actuators

The actuators are used for executing the contextual requirements' expected behaviors defined in Table 21. When drowsiness is detected, smart vehicles may use feedback to warn the driver. Such feedback can be, for example, a warning sound, voice, light or vibration [114]. When driver's drowsiness level reaches high values, sophisticated actuators such as lane keeping systems may support them better. Below, we list the set of actuators we have considered in SACRE:

– ***Seat-vibration alarm***. Graded seat-vibration alarm has been perceived as a trusted (and less annoying) actuator for warning drivers [113]. In the evaluation of SACRE, we have considered a *seat-vibration alarm* (circles in Figure 36a) based on the prototype presented by Lee et al. [113], for supporting drowsy driver context (see cr_1 in Table 21).

– ***Sound-light alarm***. Substantial research shows that complementing visual cues with redundant cues in another sensory mode speeds people's reaction time. A common

combination of sensors is the use of visual and auditory displays [113]. Thus, in the evaluation of SACRE, we have used a *sound-light alarm* actuator (squares in Figure 36a) for alerting, dangerously drowsy drivers (see cr_2 in Table 21).

– ***Lane keeping support***. In order to prevent accidents caused by fatigue drivers, many researchers have focused on studying the self-driving functionality of smart vehicles. Diverse lane keeping assistance systems and lane departure avoidance systems have been proposed [113]. These systems are characterized by being punctually activated when critic situations occur (e.g., when a driver falls asleep). We consider a *lane-keeping support* system for responding to the *driver is sleeping* context (see cr_3 in Table 21). Examples of such systems can be found in the contributions of BMW [118] and Toyota Motor Sales [119]. Figure 36b provides an illustration of this actuator.

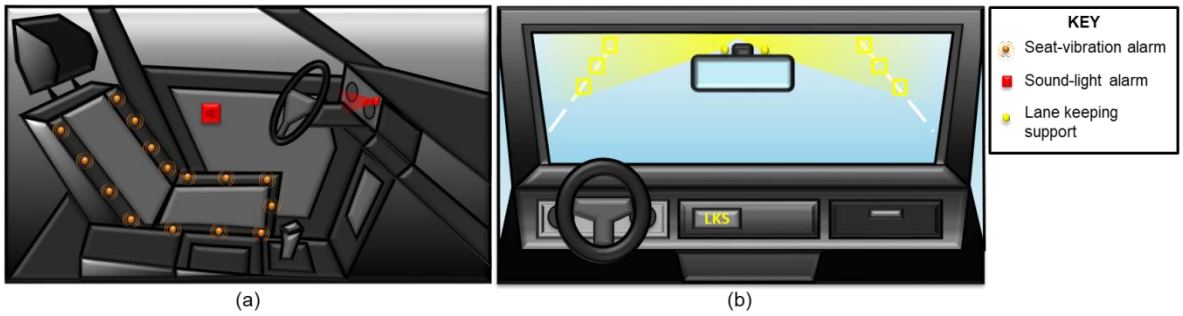


Figure 36: Smart vehicle actuators: (a) seat-vibration and sound-light alarm, (b) lane keeping support system

Actuators can be turned off by the driver after they are activated by the system (for satisfying contextual requirements), or disabled when the driver wants to keep them off. Particularly, the lane keeping support actuator can also be turned on by the driver. The actions of turning on/off and disabling actuators, triggered by the driver, result in candidate adaptations of the SAS's contextual requirements stored in the AM's knowledge base. The adaptation of will lead the smart vehicle to activate an actuator at some point of time, while in the past it was not the case, or the other way around.

4.4.2 Implementation of SACRE

The modules of the SACRE implementation are shown in Figure 37. The implementation has can be split into two main parts:

PART 1 – Self-improvement property

– ***Top-layer MAPE-K loop***. This module implements the AM in charge of the adaptation of the middle-layer loop. The MAPE-K loop elements (i.e., monitor, analyze, plan, execute, knowledge base, sensors and effectors) as well as the modules in charge of managing their policies have been implemented in Java ME 8.1. The actual policy documents were created at design-time as *.properties* files. Asynchronous communication between the Monitor and the

Analyzer and between the Monitor and the Knowledge base modules was implemented with the help of buffers. This decision was done based on previous experiences, in which we have noticed that these modules are the most work-intensive. The Knowledge base module stores context data in the form of *.arff* files in the File System for being later used by the Data Mining component.

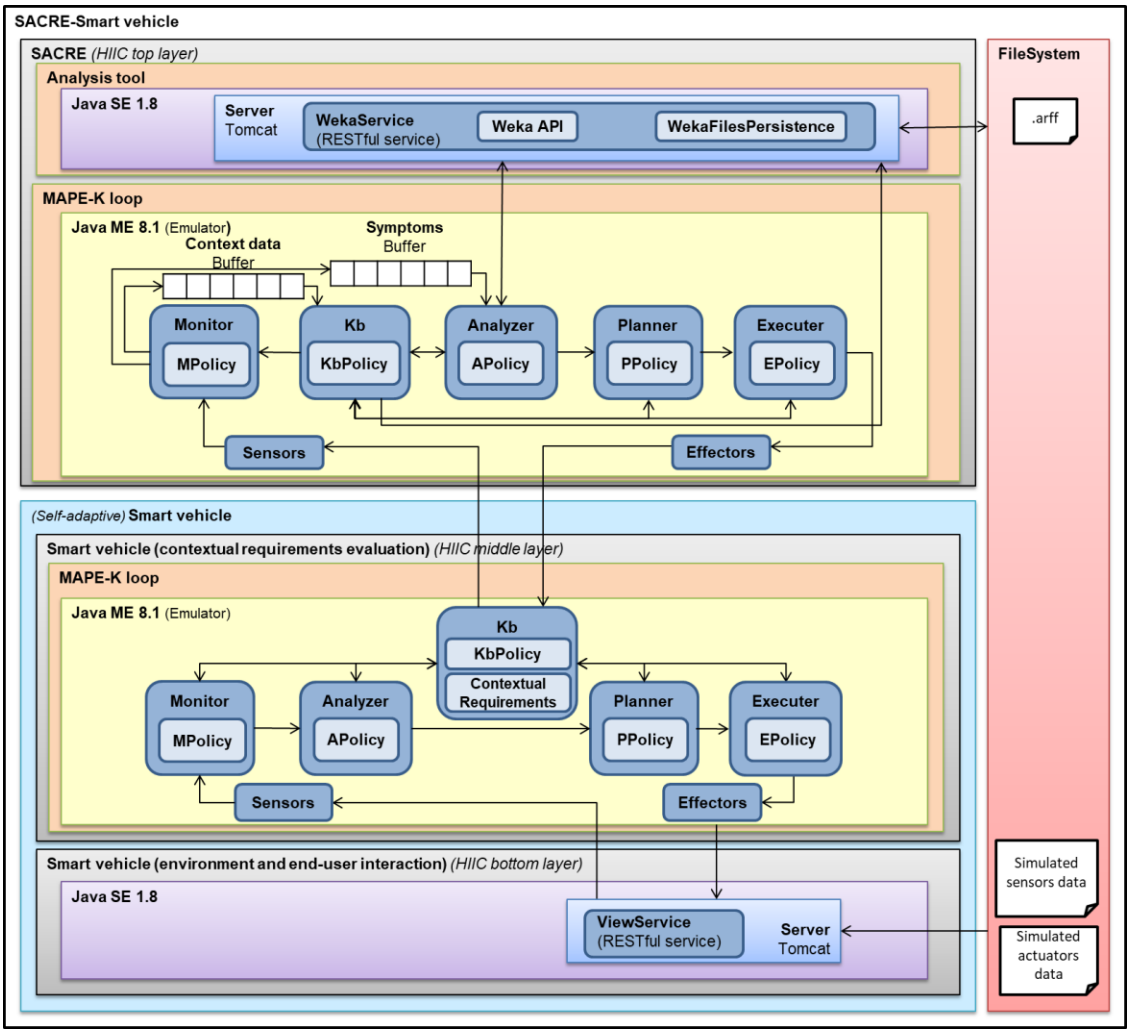


Figure 37: Implementation of SACRE for the smart vehicles domain

– ***Analysis tool***. SACRE utilizes Data Mining for finding patterns on runtime data and adapt contextual requirements' operationalization based on those patterns. For the evaluation of SACRE, the JRip algorithm [38], [39] and the Weka tool [40] have been used for applying Data Mining at runtime. The Analysis tool component is in charge of using the Weka tool Java API for applying the JRip algorithm. It also supports the Knowledge base in the persistence of the *.arff* files, in the File System. Since Java ME was not compatible with the Weka API, we have used Java SE (Java Platform, Standard Edition) 1.8 for implementing this

component. In order to enable the communication between the Analysis tool and the MAPE-K components, the tool has been implemented as a RESTful service.
The *.arff* format is the format used by Weka and it consists in a file with a header and a body. The header contains the list of the context variables and their type, as well as the variable to analyze. The actual context data is then provided in the body. The separation of the Analysis tool and the MAPE-K loop modules has allowed us to reuse the implementation of the first one in the subsequent evaluations of HAFLoop.

PART 2 – Smart vehicle

- *Middle-layer MAPE-K loop*. The feedback control loop in charge of the contextual requirements evaluation has been implemented in Java ME 8.1 by a single module. For the sake of simplicity, in Figure 37 we show this module as separated MAPE-K elements. The policies, including the initial set of requirements, are provided to the AM as *.properties* files. Requirements are loaded in memory as runtime variables, and when an adaptation is received, these variables are updated.
- *Vehicle logic*. Finally, a module simulating the interaction of the driver with the smart vehicle has been implemented. This module reports sensors and actuators data. The module has been implemented in Java SE 1.8. The communication with the middle-layer MAPE-K loop is done through a RESTful interface. Sensors and actuators data is simulated and read by the smart vehicle view module (see Figure 37), from the File System. These files have been created at design-time.

The source code of this implementation as well as more details about its construction, artifacts, and instructions of usage are available at https://github.com/edithzavala/sacre-sv.

4.4.3 Evaluation of SACRE

The evaluation of SACRE aimed at assessing the feasibility of adding self-improvement capabilities to modern SASs. The smart vehicles domain is extremely demanding in terms of both functionality and response time. Therefore is a perfect example for testing the fundamental ideas of our proposal, contained in SACRE. The evaluation of SACRE has been performed in real-time using a simulated environment. Concretely, we have used an Intel^{R} Core^{TM} 2 Duo CPU P7350 @ 2.00 GHz, with 3,0GB RAM for running the evaluation. In the remainder of this section, we describe the evaluation process and the threats to validity we have identified for this evaluation.

► Preparation activities

In Table 22 and Table 23, we provide the policy variables' values we have set for the top and middle-layer loops, respectively.

Table 22
Top-layer MAPE-K loop policies

Policy	Configuration variable	Value
Monitor	Monitoring variables	perclos, facePosition, heartBeatsPerMinute, handsOnSteeringWheel
	Monitoring variables normalization max	perclos Max = 100, facePositionMax = 1 heartBeatsPerMinuteMax = 120, handsOnSteeringWheelMax = 2
	Monitoring variables normalization min	perclos Min = 0, facePositionMin = 0, heartBeatsPerMinuteMin = 0, handsOnSteeringWheelMin = 0
	Pre-processing functions	perclos = calculate(eyesSate), facePosition = -, heartBeatsPerMinute = -, handsOnSteeringWheel = -
	Monitoring variables min	perclos Min = 0, facePositionMin = 0, heartBeatsPerMinuteMin = 0,3, handsOnSteeringWheelMin = 0
	Monitoring variables max	perclos Max = 1, facePositionMax = 1 heartBeatsPerMinuteMax = 1, handsOnSteeringWheelMax = 1
Analyzer	Analysis variables	perclos, facePosition, heartBeatsPerMinute, handsOnSteeringWheel
	Data Mining algorithm	JRip
	Data Mining tool	Weka
	Data Mining expected output	Rules, Precision, Recall, fMeasure
	Min analysis iterations	N/A. *This parameter has been indicated in source code. Since no experimental evidence is available for setting this parameter we configure it as 0 iterations (except for uncertainty case 2(a, b and c) in Table 20 for which we set 3 iterations because in that case Data Mining is not used and we want to avoid requirements' adaptation triggered by isolated uncertainty situations)*
Planner	Data Mining measures	Precision, Recall, fMeasure
	Data Mining measures min	PrecisionMin = 0,95, RecallMin = 0,95, fMeasureMin = 0,95
Executer	Managed element(s)	N/A. Indicated in code (smart vehicle)
Knowledge base	Loop frequency	14,28 iterations per second. *By experience, this is the highest frequency rate SACRE can reach in the machine used for the evaluation. Authors from previous work* [120] *in the domain of*

		smart vehicles, determined that a minimum rate of 5-10 iterations per second is required for correctly detecting drivers fatigue. Thus, this frequency value is good and supported by the literature.
	Min uncertainty iterations	3 iterations. *There is no evidence for setting this parameter, thus we inverted the analysis policy configuring 3 iterations for the uncertain cases requiring Data Mining and 0 (indicated in code) for uncertainty situations of case 2 (a, b and c in Table 20).*
	Variables to persist	perclos, facePosition, heartBeatsPerMinute, handsOnSteeringWheel, cr1ExpectedBehaviorState, cr2ExpectedBehaviorState, cr3ExpectedBehaviorState

Table 23
Middle-layer MAPE-K loop policies

Policy	Configuration variable	Value
Monitor	Monitoring variables	eyesState, facePosition, heartBeatsPerMinute, handsOnSteeringWheel
	Monitoring variables normalization max	eyesStateMax = 1, facePositionMax = 1 heartBeatsPerMinuteMax = 120, handsOnSteeringWheelMax = 2
	Monitoring variables normalization min	eyesState Min = 0, facePositionMin = 0 heartBeatsPerMinuteMin = 0, handsOnSteeringWheelMin = 0
	Pre-processing functions	N/A. Indicated in code (perclos = calculate(eyesState))
Analyzer	Contextual requirements' context	ctx1: Driver is drowsy ctx2: Driver is dangerously drowsy ctx3:Driver is sleeping
	Contextual requirements' context operationalization variables	var1: perclos var2: facePosition var3: heartBeatsPerMinute (hbpm) var4: handsOnSteeringWheel (hosw)
	Contextual requirements' context operationalization	ctx1Oper = perclos>=0,15 AND hbpm<=0,60 AND hbpm >=0,56 ctx2Oper = perclos>=0,21 AND facePosition=1 AND hbpm<=0,55 AND hbpm>=0,46 ctx3Oper = perclos >0,30 AND facePosition=1 AND hbmp<=0,45 AND hosw <1
Planner	Contextual requirements'	beh1: Activate Seat Vibration, beh2: Activate Sound/Light Alert

	expected system behavior	beh3: Activate lane keeping support
	Contextual requirements	cr1: ctx1, beh1 cr2: ctx2, beh2 cr3: ctx3, beh3
Executer	Managed element(s)	N/A. Indicated in code (adaptive vehicle)
Knowledge base	MAPE-K loop frequency	20 iterations per second
	Contextual requirements' operationalization update functions	ctx1Oper = process(ctx1SACRENewOper) ctx2Oper = process (ctx2SACRENewOper) ctx3Oper = process (ctx3SACRENewOper)

In SASs, runtime uncertainty can be caused by different factors. In the work of Knauss et al. [61], four main cases were identified (see Table 24).

Table 24
Uncertainty cases affecting SASs' contextual requirements' satisfaction

Case	*Detection of uncertainty*
Case 1	No operationalized context.
Case 2 a)	Sensor lost.
Case 2 b)	Sensor de-calibrated.
Case 2 c)	Sensor up (again).
Case 3	Violation (i.e., requirement's context holds (true) but expected behavior is not active (false)).
Case 4	Potentially wrong context (i.e., requirement's context does not hold (false) but expected behavior is active (true)).

In order to evaluate SACRE, we have designed six uncertainty scenarios (**us_1** to **us_5** in Table 21). Each scenario focuses on a specific uncertainty case (from Table 20) that at certain point (determined by the number of iterations in Table 25) affects one or more contextual requirements (from Table 17), triggering the adaptation of them.

▶ Scenarios execution

In order ensure the reliability of the results, we have replicated several times the execution of the uncertainty scenarios. For calculating a correct number of replications, we have used the formula of Berenson and Levine [121]:

$$n = \frac{Z_{\alpha}^{2} N p q}{e^{2}(N-1) + Z_{\alpha}^{2} p q}$$

Table 25
Uncertainty evaluation scenarios

Id	*Uncertainty case*	*Loop iterations*
us_1	cr1 affected by uncertainty case 3 *(vibration alarm disabled)*	1.000
us_2	cr2 affected by uncertainty case 3 *(sound-light alarm disabled)*	15.000
us_3	cr3 affected by uncertainty case 3 *(lane keeping support disabled)*	30.000
us_{4a}	cr2 and cr3 affected by uncertainty case 2b *(driver's vigilance level camera sensor reports facePosition variable values out of thresholds)*	45.000
us_{4b}	cr1, cr2 and cr3 affected by uncertainty case 2c *(driver's vigilance level camera sensor reports facePosition variable values within thresholds again)*	60.000
us_5	cr3 affected by uncertainty case 4 *(lane keeping support manually activated)*	75.000

Where:

- n: is the number of replications we require (i.e., sample size).
- Z_α: is the value from the standard normal distribution for a selected confidence level. We selected a typically used 95% of confidence level which corresponds to a Z_α of 1,96.
- N: is the total population size. In our case, this is the total number of executions we expect for the system, i.e., the smart vehicle. Considering a vehicle lifespan of 15 years [122], and a twice-daily use, we set a total population size of 10,950 executions.
- e: is the sample error we accept for this evaluation. We considered a 0,1.
- p & q: are the probability of success and failure respectively. We used the typical value of 0,5 for each of them.

Given the formula and variables' values presented above, we obtained a n of 95,21 replications. Based on this result, we have decided to run 100 replications for each uncertainty scenario.

Figure 38 shows the normalized values of the sensors' variables in each uncertainty scenarios. The x-axis of each sub-graph shows the number of iteration while the y-axis shows the normalized value. On the other hand, Figure 39 shows the actuators' state (0 for inactive, 1 for active) during the execution of each scenario. The x-axis of each sub-graph shows the number of iteration while the y-axis shows the state. Moreover, in Figure 38 and Figure 39 the exact time at which the uncertainty cases are experienced can be seen. The resulting adapted contextual requirements' operationalization per scenario is presented in Table 26. In bold, we present the variables that have been adapted, i.e., changed or added. In the cases of the variables that have been removed, we simply do not include them in the new operationalization. Since we have used a statistical method for determining new operationalizations, in some cases, different valid operationalizations may results in different replications, such it was the case of us_5.

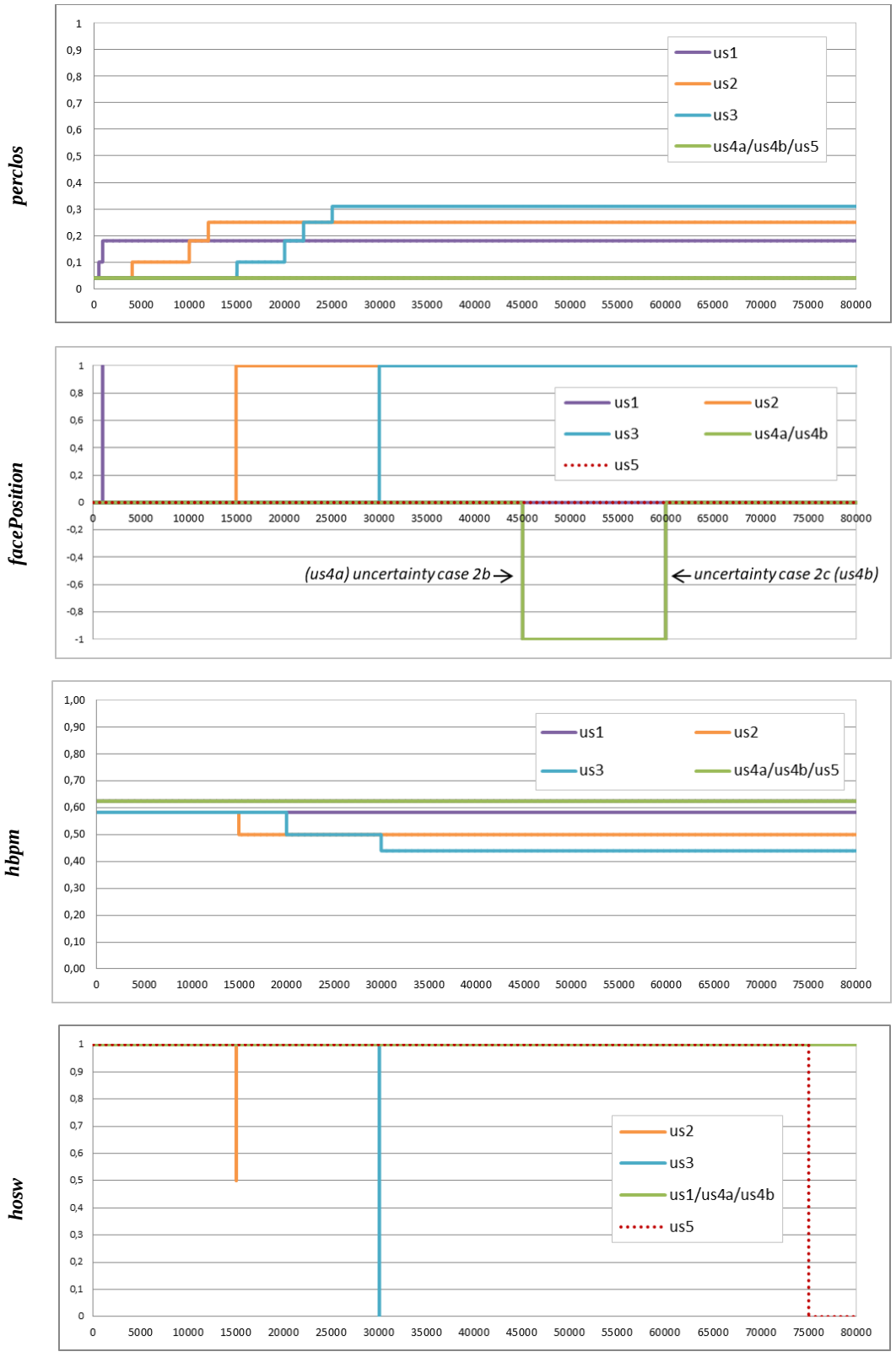


Figure 38: Sensors' variables values over execution time

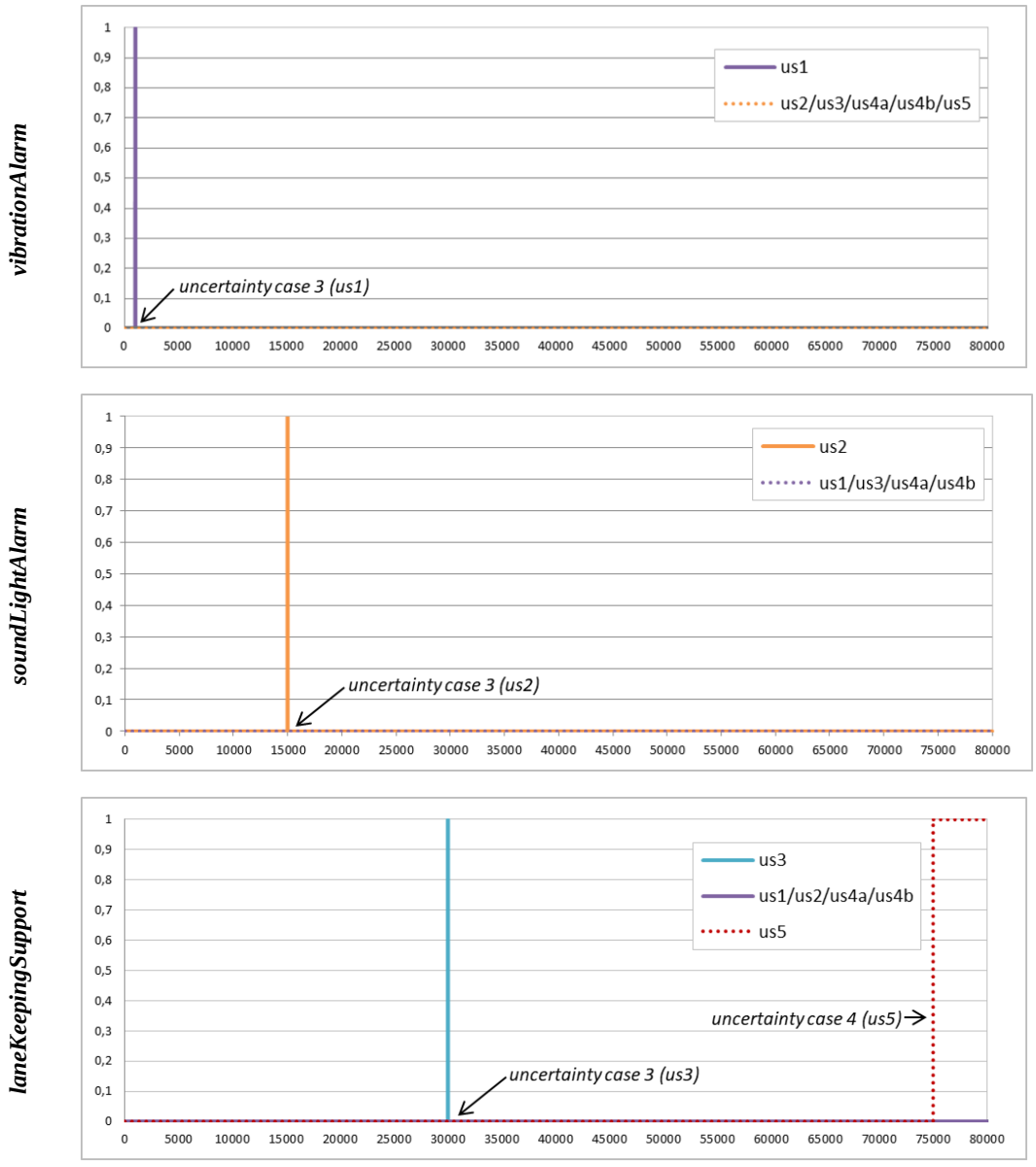


Figure 39: Actuators' state over execution time

Analysis of the results

In order to analyze the results we have: 1) explored response time (time elapsed since the uncertainty case is experienced until the requirements' adaptation is enacted), 2) statistically assessed the Data Mining algorithm.

– ***Response time results:*** In Table 27, we provide the replications' average response time (in milliseconds) for each of the uncertainty scenarios executed. We include for each average response time its standard deviation.

Table 26
Resulting adapted contextual requirements

Uncertainty scenario	CR adapted	Initial operationalization	Adapted normalized operationalization
us_1	cr_1	perclos>=0,15 AND hbpm<=0,60 AND hbpm >=0,56	perclos>=0,15 AND hbpm<=0,60 AND hbpm>=0,56 **AND facePosition=1**
us_2	cr_2	perclos>=0,21 AND facePosition=1 AND hbpm<=0,55 AND hbpm>=0,46	perclos>=0,21 AND facePosition=1 AND hbpm <=0,55 AND hbpm >=0,46 **AND hosw=0,5**
us_3	cr_3	perclos >0,30 AND facePosition=1 AND hbmp<=0,45 AND hosw <1	perclos >0,30 AND facePosition=1 AND hbpm <=0,45 AND hosw**=0**
us_{4a}	cr_2	perclos>=0,21 AND facePosition=1 AND hbpm<=0,55 AND hbpm>=0,46	perclos>=0,21 AND hbpm <=0,55 AND hbpm >=0,46
	cr_3	perclos >0,30 AND facePosition=1 AND hbpm<=0,45 AND hosw <1	perclos >0,30 AND hbpm <=0,45 AND hosw<1
us_{4b}	N/A	N/A (facePosition variable added to the active variables set for future operationalizations)	N/A (facePosition variable added to the active variables set for future operationalizations)
us_5	cr_3	perclos >0,30 AND facePosition=1 AND hbpm<=0,45 AND hosw <1	perclos **<0,05** AND facePosition**=0** AND hbpm <=**0,75 AND hbpm>=0,56** AND hosw<1 *OR* perclos **<0,05** AND face position**=0** AND hbpm <=**0,75 AND hbpm>=0,56** AND hosw**=0**

Table 27
Average response time per uncertainty scenario

Uncertainty scenario	Average adaptation response time (ms)	Adaptation response time standard deviation (σ)
us_1	3.859,50	1.004,60
us_2	9.271,46	2.229,70
us_3	13.358,25	3.028,38
us_{4a}	2.477,52	689,39
us_{4b}	261,63	124,31
us_5	30.262,09	5402,64

Figure 40 presents the detailed response time values obtained in each of the replications of the six uncertainty scenarios, represented each in a separate sub-graph. The *x*-axis of each sub-graph shows the number of replication, while the *y*-axis the adaptation response time.

The response time values obtained in the different uncertainty scenarios go in average: from 3,85 sec. to 30,26 sec., for scenarios where Data Mining was required (all, except **us4a** and **us4b**); and from 0,26 sec. to 2,47 sec., for scenarios that do not require Data Mining. For the first type of scenarios, graphs in Figure 40 suggest a correlation between the amount of data to analyze (dictated by the number of loop iterations elapsed before triggering the uncertainty case, see Table 25) and the experienced adaptation response time. Thus, we have calculated the Pearson Product-Moment Correlation Coefficient (PPMCC) for the scenarios' iterations

and the corresponding average response times. We have obtained a coefficient of 0,99 which corroborates the existence of a correlation.

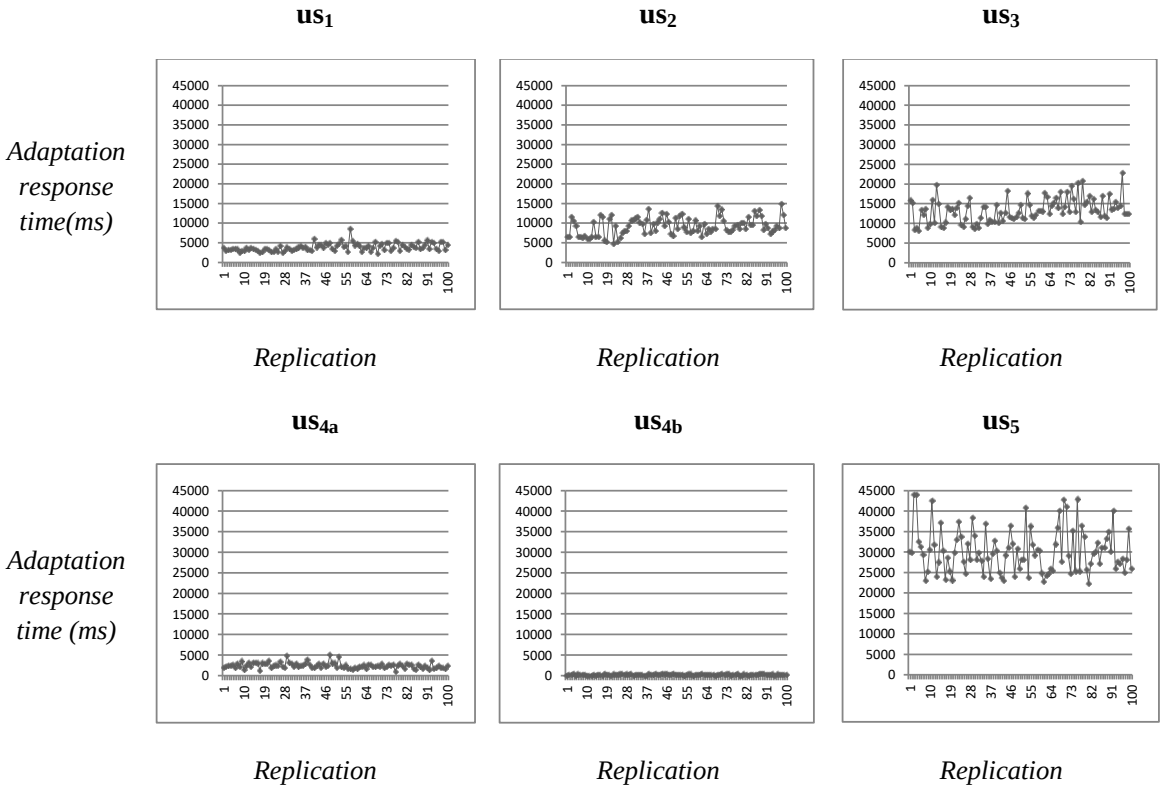


Figure 40: Adaptation response time per uncertainty scenario replication

– *Statistical analysis:* In order to analyze the performance of the Data Mining module, we have used 10-fold cross validation. The cross validation has been executed at runtime every time the data algorithm was called. If the resulting *precision*, *recall,* and *f-measure* are above the acceptance thresholds indicated in the Planner element, requirements' adaptation is accepted. We present in Table 28 the average resulting values of precision, recall, and f-measure in each of the uncertainty scenarios. We have average the values reported in each scenario replication when an adaptation is accepted. Uncertainty scenarios us4a and us4b are not included in the table since they did not use the Data Mining algorithm.

Table 28
Data Mining algorithm measures

Uncertainty scenario	*Precision*	*Recall*	*f-measure*
us_1	1	1	1
us_2	1	1	1
us_3	1	1	1
us_5	0,969059	1	0,984252

Figure 41 provides the details about the resulting Data Mining measures in each of the replications of the six uncertainty scenarios, represented each in a separate sub-graph. The *x*-axis of each sub-graph shows the number of replication, the *y*-axis the measure precision, recall or f-measure accordingly. As it can be noticed, the results of the measures were very high: invariantly 100%, for each of the measures in uncertainty scenarios **us1**, **us2** and **us3**; and, 96,90%, 100% and 98,42%, for the precision, recall and f-measure respectively for the uncertainty scenario **us5**. Variations in measures values, when existent, are presumably very small, thus we have not statically analyzed them.

It is worth to mention that particularly, in the uncertainty scenario **us5**, the Data Mining algorithm presented an output variation between replications, generating two different, but still valid, operationalization's adaptation. The factors generating these variations could also explain the resulting measures in this uncertainty scenario. However, the study of the variation due to the internal operation of the Data Mining algorithm was out of the scope of the evaluation. For better understanding this and other Data Mining algorithms' operation we refer the reader to previous works [61], [123], [124].

Threats to validity

- ***Construct validity***. In SACRE's evaluation, a threat to construct validity was that it was based on a simulated environment in which sensors and actuators data was specifically designed. Thus, the evaluation could be affected by our interpretation of contexts and interactions of the driver with the smart vehicle. In order to reduce this threat, we have studied each of the variables simulated, in existing works of the domain, and tried to model each of the variables as closest as possible to a real behavior, independently and in conjunction.
- ***Internal validity***. The internal validity of SACRE's evaluation concerns to our ability to draw conclusions about the connections between the uncertainty scenarios and the resulting adaptation response time and Data Mining measures. In order to reduce this threat, we have quantitatively interpreted our results using descriptive statistics for determine tendencies, dispersion and dependencies.
- ***External validity***. External validity refers to the generalizability of our conclusions. SACRE has been evaluated in the domain of smart vehicles. The results were satisfactory. However, due to the simulated environment in which the evaluation has been executed, generalization may be limited, not only to the domain, but also to the application in the domain. Motivated by this fact, we have conducted a series of experiments in other execution environments, which we will present later in this thesis document.

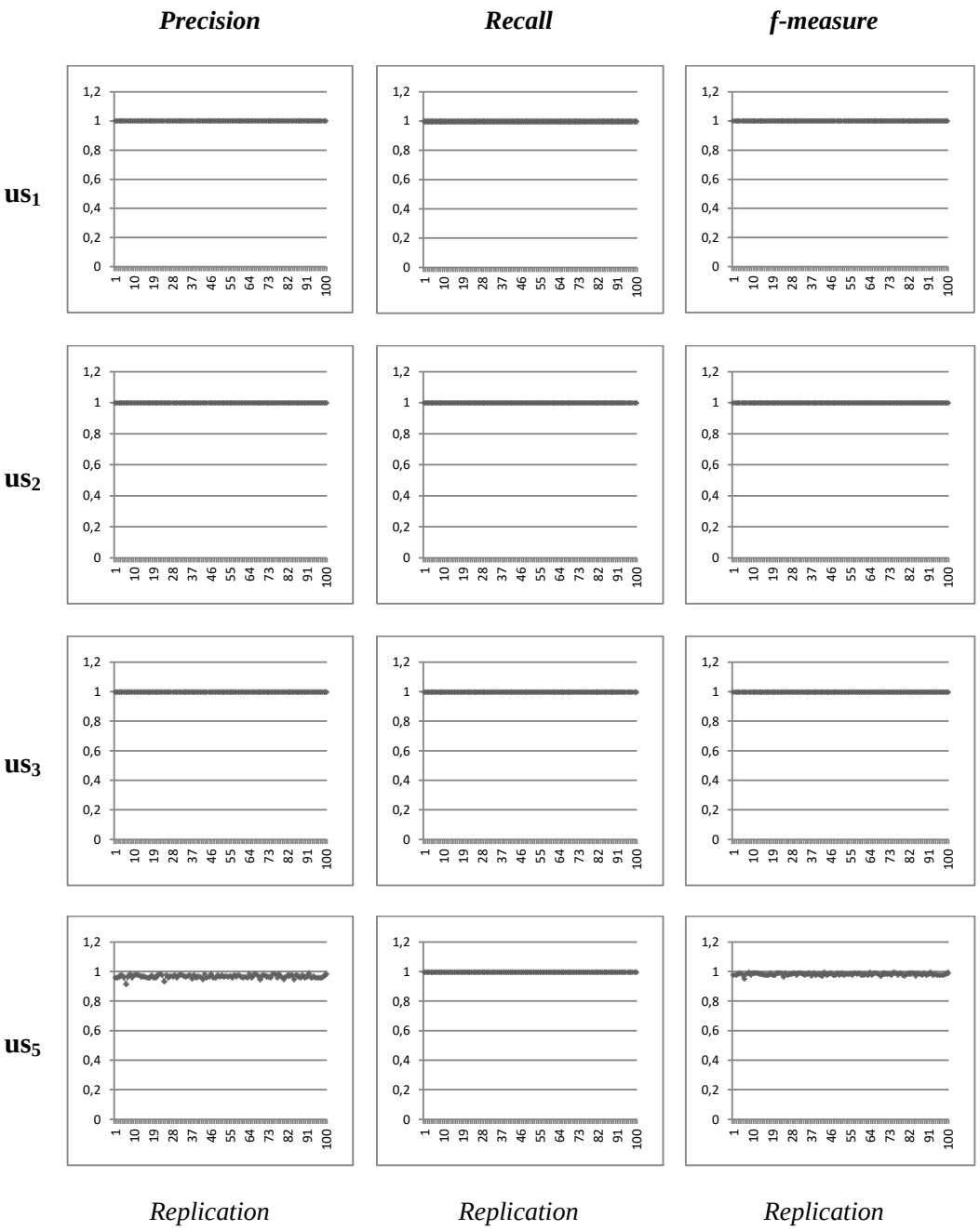


Figure 41: Data Mining algorithm measures per uncertainty scenario replication

Discussion

SACRE condenses the first ideas of our architectural proposal, HAFLoop, for correctly supporting the construction of SASs with self-improvement capabilities. The implementation of SACRE has pointed out software requirements (e.g., asynchronous communication) and reusability opportunities (e.g., analysis tool) in terms of components, and communication and data handling mechanisms, at a lower level. Moreover, the results of the evaluation of SACRE were promising regarding the feasibility of

adopting feedback loop-based architectures in demanding application domains such as the smart vehicles. Concretely, they show up the benefits of applying such kind of solution and the potential of it in the smart vehicles domain. In the evaluation of SACRE, we have focused on the adaptation of the AM for better detecting and supporting drowsy drivers. In order to demonstrate the validity of our proposal in different setting, for the evaluation of HAFLoop, we have designed a set of different use cases in which different types of adaptation are tested.

4.5 The HAFLoop4J framework

Taking into account our findings when implementing the proof-of-concept SACRE, we have implemented HAFLoop as a framework for Java-based applications (henceforth HAFLoop4J). The framework implements the generic functionality of proposal. Due to the modularity of HAFLoop, the implementation has followed a bottom-up approach, i.e., from the simplest to the most complex components while the functionalities have been developed from the most generic to the most specific. The software modules that we have developed first are shown in Figure 42. Below, we describe each of these modules:

- ***Managers and policies***. As we have explained in Section 4.3, every MAPE-K element's component consists of three subcomponents: a *Message manager*, a *Component policy manager,* and a *Component policy*. We have implemented the generic functionality of these subcomponents as follows:
  - ***Message manager***. This module manages two behaviors: normal operation and adaptation. For the normal operation, this module provides a *processMessage()* method that should be implemented by each component's subcomponent assigned to perform this task, e.g., in the *Receiver* component, the *Message processor* is in charge of extending this *Message manager* module implementing the *processMessage()* method with ad-hoc logic. For the adaptive behavior, we have developed a *PolicyChangeListener* interface with a *listen()* method that should be implemented by all the subcomponents willing to be notified in case of a policy adaptation. The *Message manager* implements this method updating its policy variables.
  - ***Component policy manager***. This module implements a *processPolicy()* method in order to receive policy adaptation messages from the *Knowledge manager* component and communicate the changes to the corresponding *Component policy*.
  - ***Component policy***. This module implements two main methods: *updatePolicy()* and *notifyPolicy()*. The former, as its name indicates, implements the logic for updating the component's runtime policy variables. The latter, provides the logic for notifying the corresponding policy's listeners about the changes. In order to support this second method, a series of methods for managing listeners have been implemented (i.e., add, remove, update, etc.). In order to manage the whole adaptation process in the Component policy, we have utilized functional reactive programming

(http://reactivex.io/), through the implementation for Java-based programs RxJava (https://github.com/ReactiveX/RxJava) available in Java 1.8.

- ***Element component (HAFLoopElementComponent)***. This module implements two main interface methods: *doOperation()* and *adapt()*. The *doOperation()* method is the entry point of all input messages related to the components' normal operation. This method administrates input messages and calls the corresponding component's *Message manager* for processing those messages. The *adapt()* method, in its turn, receives and administrates adaptation messages sent by the *Knowledge manager* and calls the corresponding *Component policy manager* for processing those messages. Both methods use the *RxJava* library for managing and dispatching messages. Moreover, each of these methods operates in a different thread, improving components' performance.

  The generic *HAFLoopElementComponent* module also implements a series of methods for managing its recipients (i.e., add, remove, update, etc.) which is also reflected on its policy. This means that the list of a component's recipients can be adapted at runtime, resulting on element's structural changes, as we have explained before in Section 4.3. Finally, this module implements a method called *construct()* in which the corresponding subcomponents are subscribed to the *Component policy,* i.e., the element is constructed, and a first *notify()* is executed with the initial configuration. This first *notify()* is treated as any other subsequent *notify()*, normally triggered by adaptation requests.
- ***Adaptive MAPE-K element (HAFLoopElement)***. Among the most relevant methods implemented by this module, there is a *construct()* method in charge of creating the connections among the element's components and assigning them their corresponding policy (this specific implementation of an element considers one component of each type). Moreover, this method calls the *construct()* method of the components, described before. The *HAFLoopElement* module also implements a series of methods for managing element's recipients (directly linked to the element's *Sender* component) and a method for enabling other systems (e.g., other elements or MEs) to communicate with its *Receiver* component.
- ***Adaptive AM (AutonomicManager)***. Finally, similarly to the previous modules, the *AutonomicManager* module provides a *construct()* method for creating the connections among the elements of the loop, assigning them their policies and triggering their *construct()* methods. Moreover, it has an *addME()* method for connecting MEs to the loop at runtime. The *addME()* method triggers elements policies' changes which are managed with the same mechanism as the adaptations. Finally, it provides an *adaptLoopElement()* method in order to receive and enact elements' adaptations. This last method can be omitted if adaptations are managed in a decentralized way. We provide an instance of an AM called *SimpleAutonomicManager*, which implements the *AutonomicManager*'s methods and is composed of one MAPE-K element of each type.

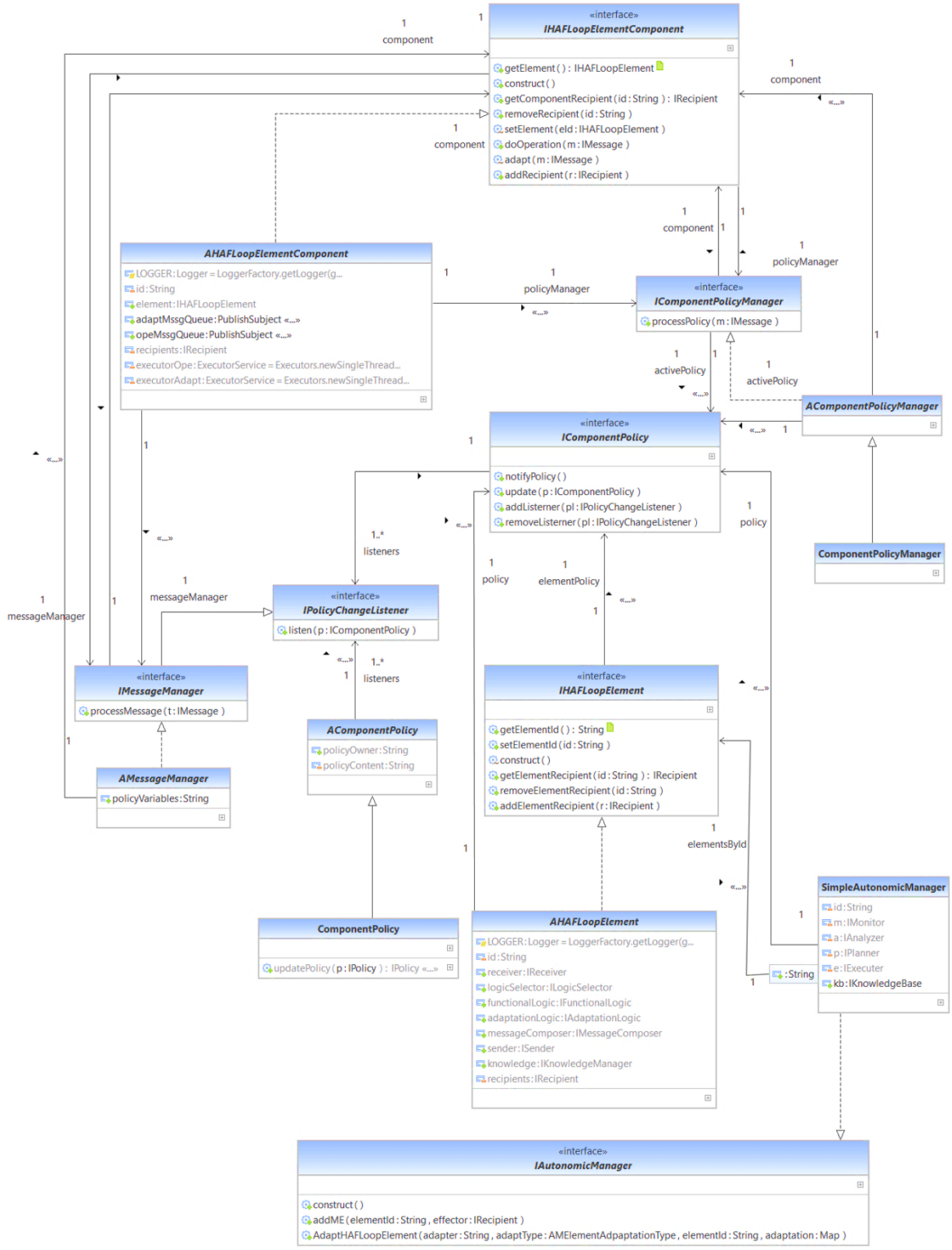


Figure 42: HAFLoop4J framework's generic modules class diagram

Subcomponents, components, and elements are aware of which component(s), element(s), and autonomic manager(s), they belong to, respectively; this information is used by the communication mechanism that we will describe later in this section. After developing the generic modules, we have implemented the ad-hoc logic of the components' *Message managers*. Figure 43 shows a class diagram of these modules. As mentioned in Section 4.3, the logics of the Functional logic component should be implemented by each HAFLoop4J instance, as well as the logic of the *Sender*'s *Message manager*, the *Message sender* module in Figure 43.

The implementation of the specific elements, i.e., Monitor, Analyzer, Planner, Executer and the Knowledge base, do not have any further functionality but the one provided by the generic element implementation described before. The same happens with the AM. We have added a *Sensor* and *Effector* interface to the Monitor and the Executer, respectively. Each HAFLoop instance should implement these interfaces for the specific use case. Systems' owners may implement as many sensors and effectors as required. Finally, for communicating internally (and externally if desired), we have proposed a standard message format (see class diagram in Figure 44). In HAFLoop, a *Message* contains the following data:

- ***To***. This field indicates the immediate next recipient of a message.
- ***From***. This field indicates the immediate previous sender of a message.
- ***Code***. This field is used for determining, based on policies, how a message should be processed (or forwarded) by the components till reaching the final recipient, e.g., if a message code indicates that a message is an adaptation, then the message should go to the adaptation logic of an element. In this implementation of HAFLoop, messages' codes are mainly used by element's components.
- ***Type***. This field indicates the type of message, e.g., request for analysis, response to a ME, etc. In this implementation of HAFLoop, the type field is mainly used for leading messages among elements and to/from MEs.
- ***Body***. This field contains the body of the message. In this implementation of HAFLoop, the message body consists of a set of key-value pairs, completely customizable for sending any type of content. In this implementation, we provide an example of a policy's adaptation message body (containing the policy owner and the new policy) and an example of a normal operation message body (containing the type of data, e.g., monitoring data, and the actual data).

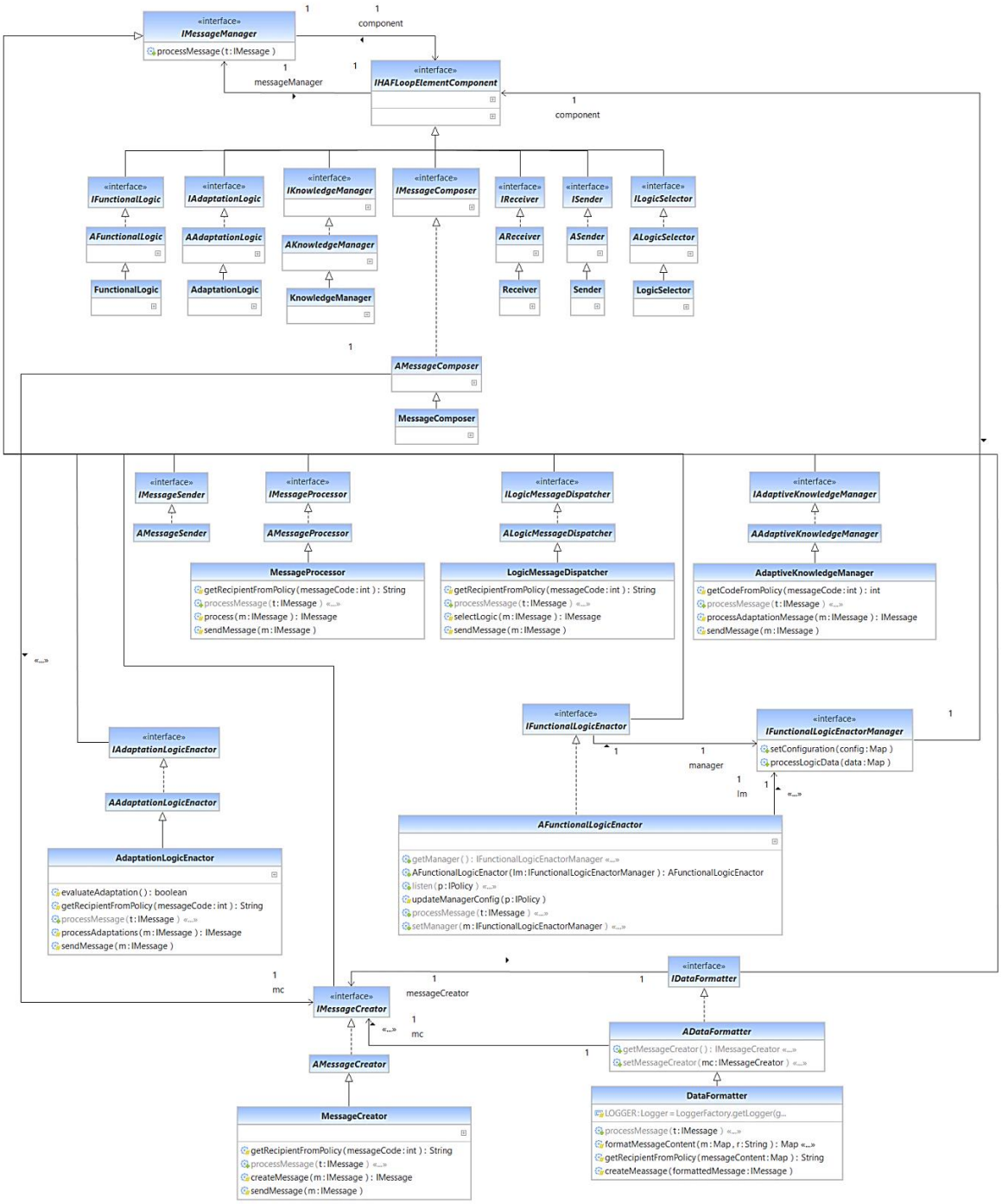


Figure 43: HAFLoop4J framework's components and ad-hoc modules class diagram

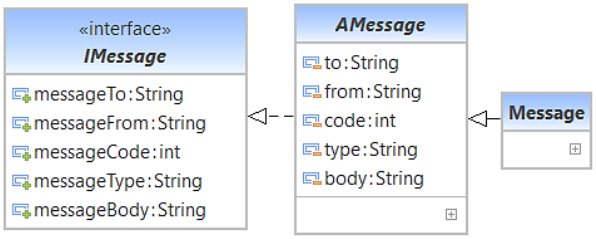


Figure 44: HAFLoop4J framework's message class diagram

The HAFLoop4J framework improves the development process of adaptive feedback loops for SASs in different ways. First, the great majority of the components and subcomponents as well as the communication mechanism, can be reused by any SAS. Therefore, systems' owners can focus on domain or application-specific issues, i.e., the development of MAPE-K elements' functional logics. Second, the operation of the components has been optimized based on previous experiences, utilizing popular software engineering techniques such as multi-threading and asynchronous communications. Third, as it is shown in Figure 43 and Figure 44, when possible, HAFLoop4J components have been structured into layers, i.e., they have an interface, and abstract class and an implementation class. Therefore, components and subcomponents can be replaced by other implementations and/or extended for fulfilling specific SASs' requirements.

Moreover, from an organizational perspective, since components are conceptually and technically loosely coupled, they can be developed independently, e.g., by different specialized teams/companies, and gradually improved as required. This characteristic is quite convenient since nowadays software systems are developed more and more in distributed environments and following agile methodologies. Fourth, regarding usability, due to the close relation between the terms typically used in the SASs' filed and the HAFLoop4J components, we consider that our framework is easy to understand, learn, and use. Finally, the source code of this implementation as well as more details about its construction, tests, artifacts, and instructions of usage are open and available at https://github.com/edithzavala/loopa.

4.6 SALI: the smart self-driving vehicle

In the last decades, many efforts have been spent on the development of self-driving (or autonomous) technology. This technology intends to replace driving tasks where the human driver is "under-challenged", for example, long distance travels on highways. Thus, the driver can focus on other tasks during such periods like doing business or relaxing. On the other hand, self-driving technology is said to be infallible from human failure because computer programs never get tired. Thus, it can manage complex and critical traffic situations where the human driver might be "over-challenged", for example, when the driver is drowsy or tired (as described in Section 4.4) [125].

Self-driving vehicles (SDVs) may face runtime challenging factors such as unpredictability (e.g., a road accident), runtime faults (e.g., a sensor fault) and limited resources (e.g., running out of battery).

While most researchers have focused on studying the self-driving functionality, less have investigated how the runtime challenging factors, mentioned before, affect and could be addressed in SDVs. Motivated by this fact, we have developed a project called SALI (SmArt seLf-driving vehIcle), funded by the Swedish program *openresearch@astazero* (https://azopenresearch.fluidreview.com/res/p/A0034/). In SALI, we have incorporated our solution, HAFLoop, to a feedback loop-based SDV, for enabling adaptation capabilities to its monitoring system. The resulting smart SDV (SSDV) is able to respond at runtime to the challenging factors mentioned before.

The evaluation of the SSDV has been performed in two environments, as part of the SALI project tasks: a simulation and a real environment. The role of the applicant in the SALI project has been as co-leader. She has coordinated and executed all the experiments with the support of technicians and engineers of the AstaZero test track and the vehicle laboratory Chalmers Revere (https://www.chalmers.se/en/researchinfrastructure/revere). Moreover, she has been in charge of implementing the software solution in both environments: the simulation and the real. In the rest of this section, we describe each of the execution environments. First, we introduce the implementation details for both cases. Second, we provide the details of the evaluation tests.

4.6.1 SALI in a simulation environment

The self-driving vehicle

OpenDaVINCI (https://opendavinci.readthedocs.io) is an open-source software environment written in C++ that acts as a middleware to realize distributed software components exchanging messages. It also provides a domain-specific library for supporting additional functions typically required by automotive software systems to realize the self-driving functionality. For instance, it provides methods to describe a logical road network, a visualization environment, and components to embody simulations (vehicle kinematics, sensor simulations for a virtual camera, infrared, and ultrasonic sensors). Moreover, it provides a series of reusable algorithms for autonomous vehicles [126]. In order to implement the adaptive feedback loop supporting our SDV, we have extended OpenDaVINCI (https://github.com/edithzavala/OpenDaVINCI) and the vehicle software environment OpenDLV (https://github.com/edithzavala/opendlv/tree/feature.smartcar).

OpenDLV (https://github.com/chalmers-revere/opendlv) is an open source software environment to support the development and testing of SDVs, both in simulation and real environments. It facilitates the migration of software modules tested in simulated vehicles to real ones. In this implementation, we have containerized software components using the Docker platform (https://www.docker.com/), and exposed them as services. In order to enrich contextual data, apart from the sensor data simulated with OpenDaVINCI, we have included data gathered through vehicle-to-vehicle (V2V) communications and a weather and traffic monitoring service. Figure 45 shows the software modules of our SDV. Below, we briefly describe each of these components:

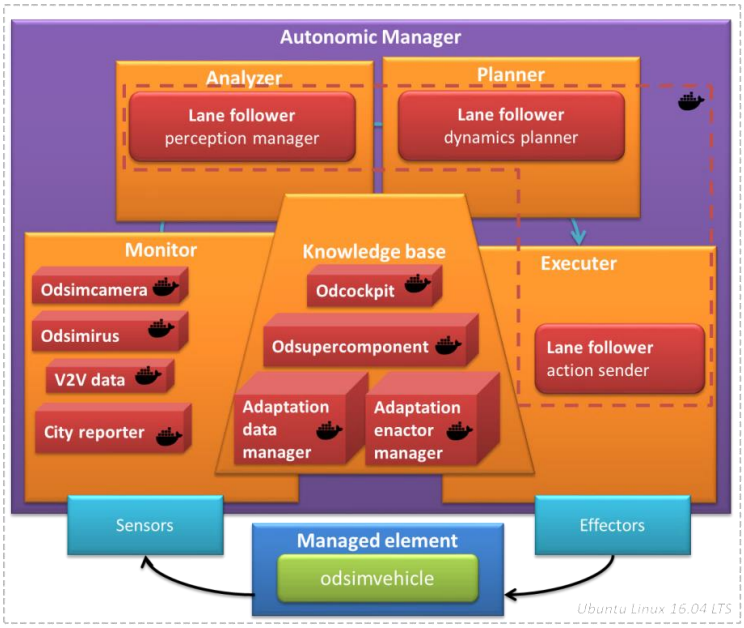


Figure 45: Self-driving vehicle (simulation environment)

SDV – OpenDaVINCI modules

- *Odsupercomponent*. This module is in charge of creating a UDP multicast session to enable the communication between the rest of components. Moreover, it provides to the components their corresponding initial configuration, i.e., policies.
- *Odsimvehicle*. This component simulates the actual vehicle, i.e., dimensions, heading, position, etc.
- *Odsimirus*. This component gathers infrared and ultrasonic sensors data from the virtual environment (see Figure 46). Three sensors of each type are simulated. Figure 47 shows the sensors layout. The Odsimirus module has been extended for supporting structural adaptation. That is, infrared and ultrasonic sensors can be (de)activated at runtime, if it is required.
- *Odsimcamera*. This component gathers camera data in the form of images from the virtual environment (see Figure 45). In the simulation, the camera captures images of the objects placed in front of the vehicle (see Figure 47). Moreover, this component provides a visualization of the video images in real-time. This specific sensor is not adaptive.
- *Odcockpit*. This component provides the visualization of the road, the vehicle, and the infrared and ultrasonic sensors (see Figure 46).

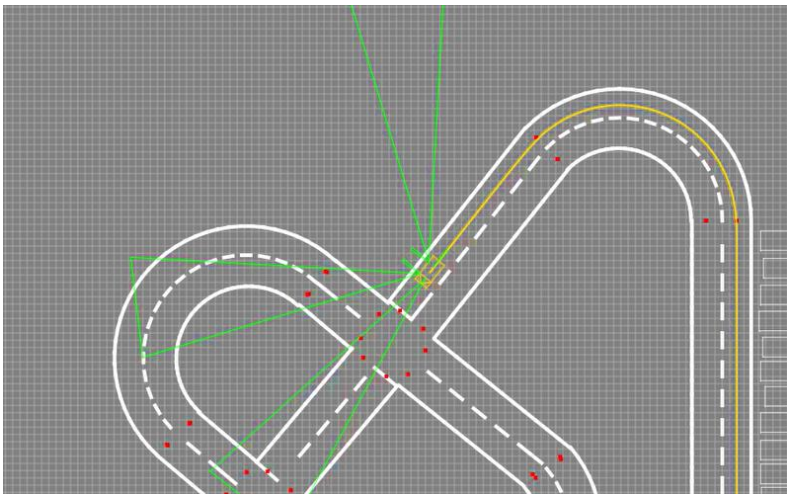


Figure 46: Self-driving vehicle's odsimirus

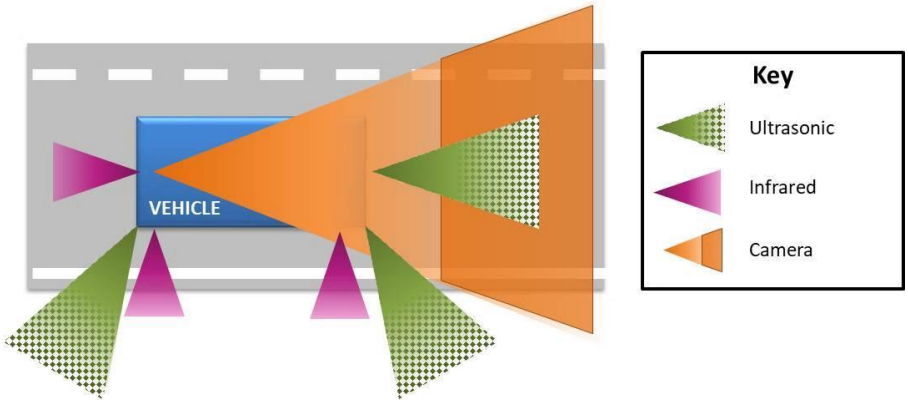


Figure 47: Self-driving vehicle's sensors layout (simulation environment) [125]

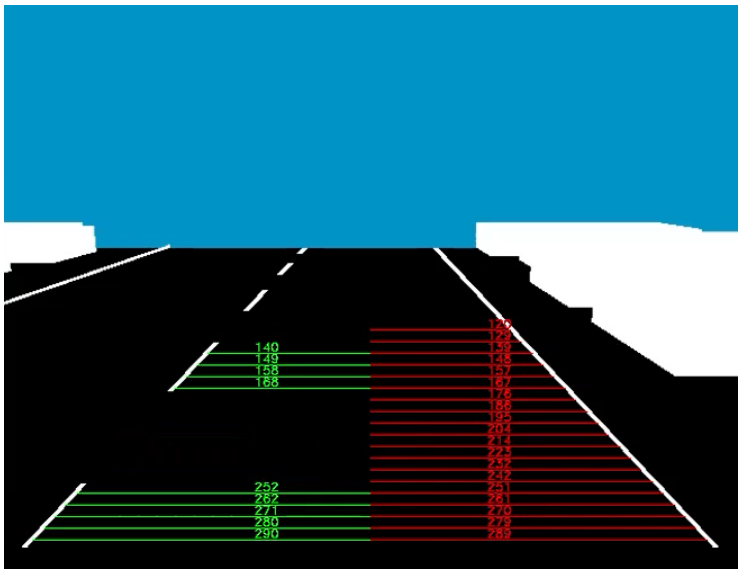


Figure 48: Self-driving vehicle's odsimcamera

- *Lane follower*. This module has been extended for supporting a context-aware SDV, i.e., a self-adaptive SDV. The driving logic performs three main tasks: 1) follow the lane, 2) overtake vehicles moving slower (or static objects), 3) recalculate route based on traffic information and road events (e.g., a crash). The logic of this component consists of three parts (even if they were implemented by a single class), corresponding to different MAPE-K elements, as it is shown in Figure 42.
  - *Perception manager*. This submodule is in charge of processing monitoring data and determining the current context (Analyzer).
  - *Dynamics planner*. This submodule processes output data from the *Perception manager*, and determines route, required acceleration, and required steering wheel angle (Planner).
  - *Action sender*. Finally, this submodule sends the proposed dynamics to the vehicle (Executer).

SDV – OpenDLV modules

- *V2V data*. This module simulates V2V communication data, both Cooperative Awareness Message (CAM) [127] and Decentralized Environmental Notification Message (DENM) [128].
- *Adaptation data manager*. This component is in charge of forwarding monitoring data gathered through sensors and services to the MAPE-K loop in charge of the monitors' adaptation (i.e., the HAFLoop instance we will describe in next subsection).
- *Adaptation enactor manager*. This component receives adaptation requests from the MAPE-K loop in charge of the adaptation of the monitors and sends requests for change to the corresponding monitors (i.e., sensors modules and monitoring services). Then, monitors update their policy variables and adapt their logic for enacting these adaptations.

SDV – External services

- City reporter. This service utilizes the API offered by HERE (https://developer.here.com/) for providing traffic load data and the OpenWeatherMap (https://openweathermap.org/) API for providing weather data. This monitoring service implements both structural and parameter adaptation.

▸ Adaptive monitoring using HAFLoop4J

In order to support the adaptation of the SDV monitoring system, we have implemented a second feedback loop using the HAFLoop4J framework. That is, two HAFLoop AMs will operate in a single SDV: a Level-1 AM implemented in C++, described before, in charge of managing the context-aware self-driving functionality; a Level-2 AM implemented in Java, in charge of Level-1 AM's Monitor adaptation. For the Java-based AM, we have imported HAFLoop4J as a library in a new project and instantiated a generic AM (concretely, we have created an instance of the *SimpleAutonomicManager* introduced in Section 4.5). Then, we have developed the functional logic and sender components of

each element. For the sake of simplicity, in Figure 49, we provide an overview of only the most relevant components of our implementation.

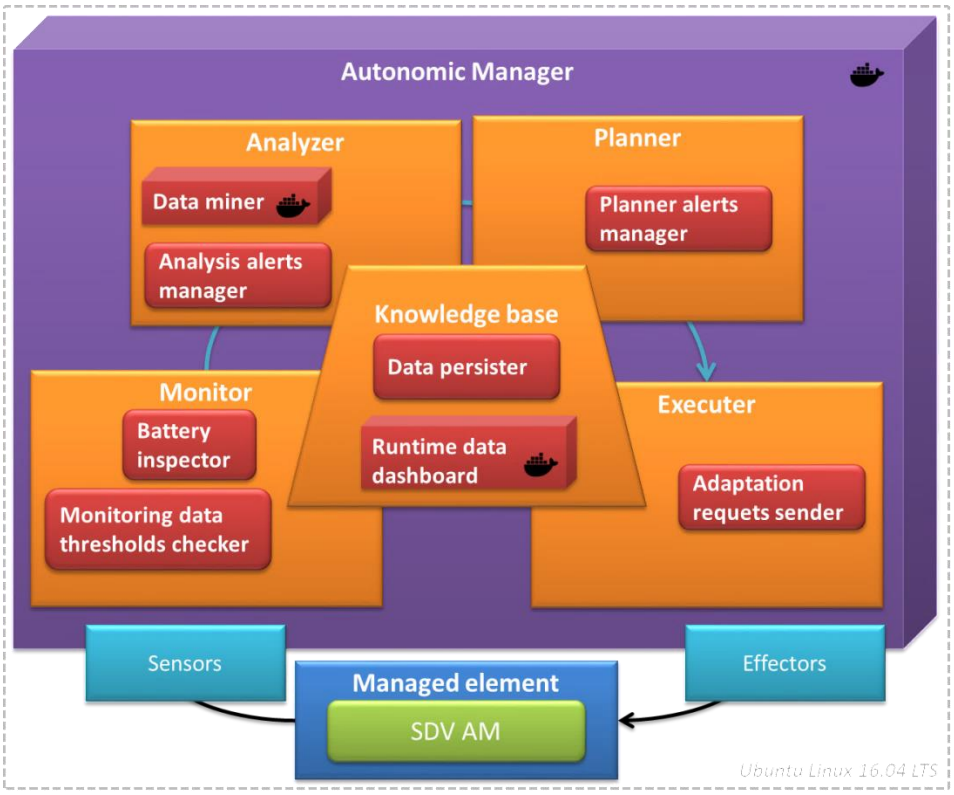


Figure 49: MAPE-K loop in charge of SDV monitors' adaptation

The MAPE-K elements are exposed as Java modules to each other, and containerized and exposed together as a service to external systems, such the SDV. The loop utilizes two external services: one for applying Data Mining techniques over monitored data (which is an extended version of the one utilized for SACRE's evaluation, see Section 4.4) and a second one for visualizing runtime data (see Figure 49). These services have also been containerized. The main modules that compose the elements of the loop in charge of SDV monitors' adaptation are:

LEVEL-2 AM – Monitor

- *Monitoring data thresholds' checker*. This component receives monitoring data and based on policies, revises its correctness. Monitoring data out of thresholds may indicate, for instance, a failure; in this case, an alert is sent to the Analyzer. In any case, monitoring data is sent to the Knowledge base for being persisted.
- *Battery inspector*. This component revises vehicle's battery level, at every loop's iteration. For this implementation, we have introduced the cost of each source of monitoring data through policies. If battery level issues are detected, an alert is sent to the Analyzer. In any case, battery level data is sent to the Knowledge base for being visualized.

LEVEL-2 AM – Analyzer

- *Analysis alerts manager*. This module receives analysis requests in the form of alerts from the Monitor. Based on the type of alert, e.g., monitor fault or a battery issue, this module analyzes historical runtime data (persisted by the Knowledge base) and determines which sensors and monitoring services will be required in the near future. In order to do that, it utilizes the Data miner service. With the Data Mining results, this module decides whether an adaptation is worth it or not. In case of yes, an alert to the Planner is sent.
- *Data miner*. The Data miner component utilizes the API of the Weka tool [40] for finding patterns on historical runtime data, in order to determine: 1) vehicles' position(s) in the near future, 2) monitors that will be required in that future position(s).

LEVEL-2 AM – Planner

- *Planner alert manager*. This component is in charge of receiving and processing alerts sent by the Analyzer. With the list of available and required monitors, it performs a trade-off between the monitors' cost, their coverage, and their utility regarding the self-driving functionality usage. Cheaper monitors are prioritized as long as they (or a combination of them) are able to gather the monitoring data required. For the most critical situations, coverage is sacrificed as long as the self-driving functionality can still be supported. As a last resort, a request for disabling the self-driving functionality is sent to the driver. A list with the monitors to adapt, and how they should be adapted, among other parameters, are sent to the Executer.

LEVEL-2 AM – Executer

- *Adaptation request sender*. This module receives adaptation requests from the Planner, decides to which ME they should be sent (in this case we only have the OpenDaVINCI SDV vehicle), transforms the requests into the required data format (understandable by the ME) and sends them.

LEVEL-2 AM – Knowledge base

- *Data persister*. This component is in charge of storing runtime data sent by the MAPE elements. Similarly to in SACRE, in this implementation, monitoring data is directly stored in .arff files, the format required by Weka. The rest of data is stored in runtime variables, e.g., active monitors, active functionalities, last adaptation request. This component is also in charge of sending received data to the *Runtime data dashboard* module, which has been independently containerized and exposed as a service.
- *Runtime data dashboard*. This module receives runtime data from the Data persister through an instance of the monitoring tool Graphite (http://graphite.readthedocs.io), and provides a visualization of this data using the Grafana (https://grafana.com/) platform (see Figure 50). For supporting the logic of this module, we have used existing Docker images (https://hub.docker.com/r/graphiteapp/graphite-statsd/ and

https://hub.docker.com/r/grafana/grafana/). Then, we have provided the corresponding configuration and designed the visualization components (i.e., graphs, alerts, etc.).

Figure 50: Level-2 AM runtime data dashboard

The source code of this implementation is available at https://github.com/edithzavala/ksam-loopa.

4.6.2 Evaluation of SALI in a simulation environment

The evaluation of SALI aimed at assessing the feasibility of supporting adaptive monitoring in modern SASs such the smart vehicles. The evaluation of SALI has been performed in real-time using the simulated environment described before. Concretely, we have used an IntelR CoreTM i7-7700HQ CPU @ 2.80GHz, with 16,0GB RAM. In the remainder of this section, we describe the evaluation process and the threats to validity we have identified for this evaluation.

▶ Preparation activities

The evaluation of HAFLoop has consisted in six use case scenarios (us_1 to us_6 in Table 29). They belong to two main use cases: sensor fault and battery level issues. Adaptation decisions are based on three main factors: the number of vehicles on the road (the more vehicles, the more increased driving risk); the typical self-driving functionality usage, learned from driver's behavior in a training phase; and, the cost and utility of each source of monitoring data, i.e., sensors, V2V communication, and cloud services.

In order to find patterns on the self-driving functionality usage, given the good results of SACRE, in SALI we have also utilized the data mining tool Weka [40]. In order to train the SSDV, we have considered a driver that goes from work to home in a daily basis and utilizes the self-driving functionality in specific segments of the journey. Models are generated offline, in a real setting it they could be generated at the end of a journey, for instance. The resulting models are then used at runtime

for: 1) predicting the position of the vehicle in the near future (i.e., next N iterations) when a fault or battery issues are experienced, 2) predicting the self-driving functionality usage in that position.

Given the nature of the data, for learning route preferences we have utilized the IBk (K-nearest neighbors) classifier [129] on vehicle's position data; meanwhile, for learning about the self-driving usage, we have utilized the JRip classifier [38], [39] on a Boolean class variable that indicated whether the functionality was active or not. The resulting rules regarding the self-driving functionality usage are shown in Figure 51.

Table 29
Use case scenarios

Id	Use case	Scenario	Expected adaptation
us_1	Sensor fault	Frontal ultrasonic sensor fails when the SSDV goes on a road with no other vehicles, at the beginning of the journey.	No adaptation is enacted. Self-driving functionality stays active.
us_2		Frontal ultrasonic sensor fails when the SSDV goes on a road with other vehicles, at the beginning of the journey.	V2V communication is activated given the increased driving risk. Self-driving functionality stays active.
us_3		Frontal ultrasonic sensor fails when the SSDV goes on a road with no other vehicles, close to the end of the journey.	No monitor adaptation is enacted. According to patterns learned, driver will change to manual mode in the near future.
us_4	Battery issues	Critical battery level is experienced when the SSDV starts its journey in a road with no other vehicles. Parameter adaptation is not supported.	A trade-off between required and non-required monitor is performed, resulting in the deactivation of the city service (traffic and weather monitoring).
us_5		Medium battery level is experienced when the SSDV is in the middle of its journey in a road with no other vehicles. Parameter adaptation is supported.	A trade-off between required and non-required data sources, and their monitoring frequency is performed, resulting in the adaptation of the traffic monitoring frequency (i.e., it is reduced).
us_6		Critical battery level is experienced when the SSDV starts its journey in a road with other vehicles.	No monitor adaptation is enacted given the increased driving risk. However, a take-over request is sent to the driver. Driver mode is changed to manual.

At runtime, Level-1 and Level-2 AMs require policy variables for driving their operation. Therefore, we have defined a set of policies for the AM elements. Table 26 and Table 27 provide a simplified version of the most relevant configuration variables considered in SALI evaluation, and the initial values assigned to them in each use case scenario. Apart from the operational-related variables listed in Table 30 and Table 31, policies also include variables related to the AMs' structure, e.g., initial list of recipients, type of messages accepted by each recipient, etc. Moreover, the Level-1 AM policies contain variables related to the simulation environment, e.g., the type of vehicle and road to use, initial vehicle position, among others.

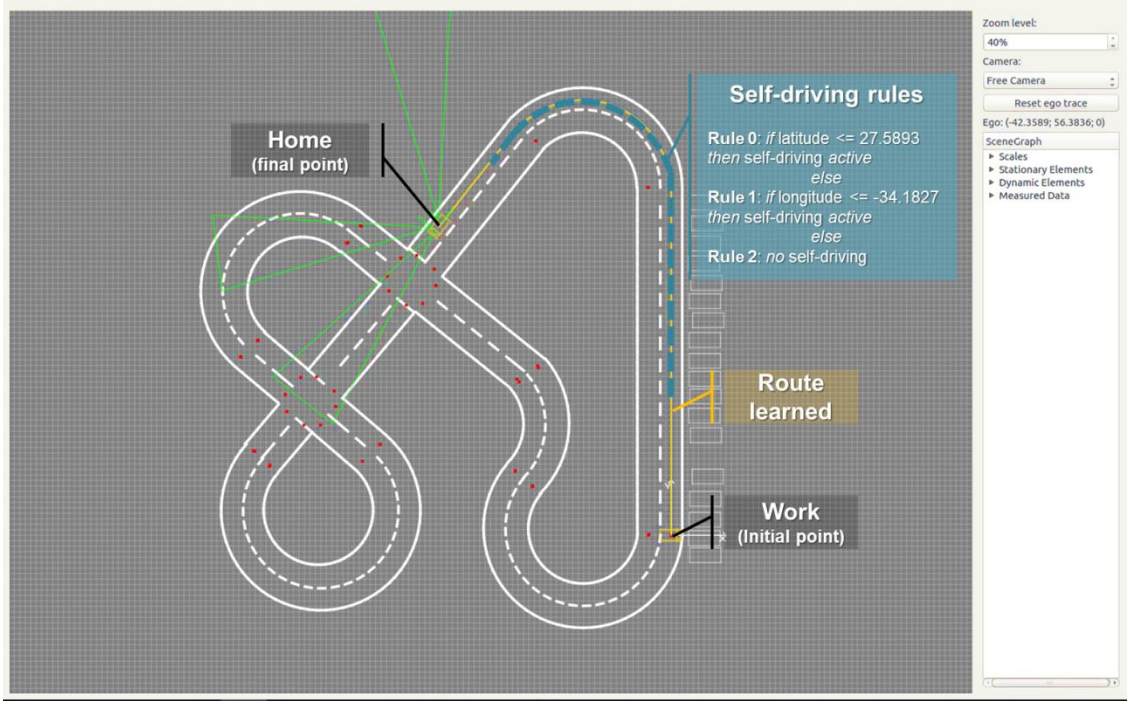


Figure 51: Self-driving patterns in the simulation environment *odcockpit*[4]

Table 30
Level-2 AM policies

Policy element	Variable	Variable description	us_1	us_2	us_3	us_4	us_5	us_6
Monitor	Alert iterations	Number of iterations, detecting sensor fault or battery issue, to wait before triggering an analysis alert	3			0		
	Initial battery level	The battery level at the initial point of each scenario execution	100%			60%		
	Battery limit	The battery level considered as critical	45%					
	Monitors	List of monitoring data sources, type of source (T), the monitoring data provided (Vars), the monitoring frequency (F) and their cost (C, a factor	**traffic**: (T) service, (Vars) traffic factor, (F) 60000ms, (C) 40 **weather**: (T) service, (Vars) weather, (F) 60000 ms, (C) 16			**traffic**: (T) service, (Vars) traffic factor, (F) 10000ms, (C) 40 **weather**: (T) service, (Vars) weather, (F) 10000 ms, (C) 16		

		in relation to the rest of monitors, taking into account power and monetary aspects, and its utility for correctly supporting the self-driving functionality)	**imu**: (T) sensor, (Vars) longitude – latitude - speed, (F) 100 ms, (C) 1 **camera**: (T) sensor, (Vars) image size – frontal distance, (F) 100 ms, (C) 10 **infrared (frontal right, rear and rear right)**: (T) sensors, (Vars) frontal right distance – rear distance – rear right distance, (F) 100 ms, (C) 5 each **ultrasonic (frontal center, front right, rear right)**: (T) sensors, (Vars) frontal center distance – frontal right distance – rear right distance, (F) 100 ms, (C) 3 each **V2V**: (T) service, (Vars) distance – road event, (F) 100 ms, (C) 30			
	Monitoring variables	Variables to be monitored and variables' values characteristics (type (T), min, max or possible values (Val)). Values out of min-max range or not listed as possible values, might indicate a monitor fault	**traffic factor**: (T) Double, (Val) -1, 10 **weather**: (T) String, (Val) Rain, Snow, Extreme, Clear, Clouds, Foggy, Fog, Drizzle, Mist **longitude**: (T) Double, (Val) -180, 180 **latitude**: (T) Double, (Val) -90, 90 **speed**: (T) Double, (Val) 0, 2 [m/s] **frontal right distance, rear right distance, frontal center distance, rear distance**: (T) Double, (Val) -1, 39 **road event**: (T) String, (Val) Crash **image size**: (T) Double, (Val) 0, 5000000 (this variable is useful for determining camera correct operation)			
	Initial monitors	Initial set of active monitoring data sources	traffic, weather, imu, camera, infrared (frontal right, rear and rear right), ultrasonic (frontal center, front right, rear right)			
Analyzer	Alert iterations	Number of iterations, receiving an alert from the Monitor, to wait before triggering Data mining analysis	3	0		
	Adaptation supported	Type(s) of adaptation supported	Structural (S)	S	Para.	S
	Analysis technique	This could include: technique, tool, endpoint, algorithms, algorithms' parameters	**Technique**: Machine Learning **Tool**: Weka **Endpoint**: protocol, host, port **Algorithms**: JRip, IBk **Parameters**: Positions to predict (N = 100)			
	ME functionalities	Critical functionalities to be provided by the ME	Self-driving			
Planner	Plan technique	This could include: technique, tool, endpoint, algorithms, algorithms' parameters	Technique: Objective function (min (monitoring cost), max (monitoring data required by the ME functionalities))			

	Monitoring data required by ME functionalities	Monitoring data required by each of the ME functionalities	Self-driving: longitude, latitude, speed, frontal right, rear right and frontal center distance
Executer	Level-1 AM endpoint	ME interface to communicate adaptation decisions	protocol, host, port
Knowledge base	Persistence format	The format in which data is going to be persisted	.arff (the format required by the Weka tool)
	Monitoring data	List of monitoring data sources to take into account for persistence	traffic, weather, imu, camera, infrared (frontal right, rear and rear right), ultrasonic (frontal center, front right, rear right), V2V

Table 31
Level-1 AM policies

Policy element	Variable	Variable description	us_1	us_2	us_3	us_4	us_5	us_6
Monitor	Monitors	List of monitoring data sources and their monitoring frequency	Same as in Table 26					
	Monitoring variables	Variables to be monitored						
	Initial monitors	Initial set of active monitoring data sources						
Analyzer	Vehicle variables	Variables resulting from vehicle's monitoring	vehicle's position and speed, frontal right, rear right, frontal center and rear distance					
	Context variables	Variables gathered from external systems	weather, traffic, road event					
Planner	Adaptation variables	Variables to include in the adaptation plan	Acceleration, steering wheel angle, driving route					
Executer	ME endpoint	ME interface to communicate adaptation decisions	vehicle id					
Knowledge base	Level-2 AM sensors endpoint	Interface to send runtime data to Level-2 AM	protocol, host, port					

Similar to SACRE, in SALI each scenario has been replicated several times for ensuring the reliability of the results. Concretely, each scenario has been executed under the same conditions 100 times. Figure 52 illustrates how each scenario has been executed, i.e., the number of vehicles on the road and the point (average point calculated after executing all scenarios' replications) at which the sensor fault or the battery issue has been experienced.

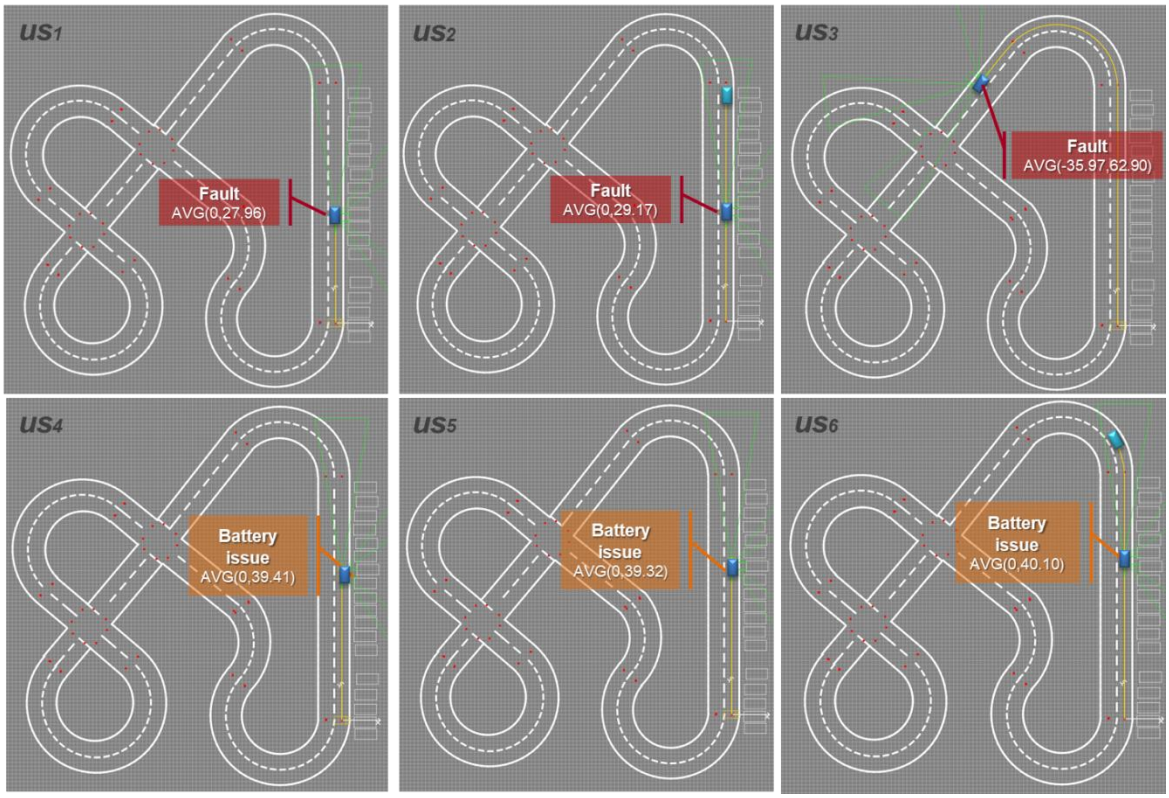


Figure 52: SALI evaluation use cases scenarios

Analysis of the results

In order to analyze the evaluation results, two aspects of the self-improvement process are explored: response time and adequacy. The response time is split into:

- *Level-2 AM response time*. Time elapsed since the Monitor element detects a sensor fault or a battery issue until an adaptation decision (in case of no adaptation required) or an adaptation request is sent to the Level-1 AM.
- *Level-1 AM response time*. Time elapsed since an adaptation request is received since it is enacted.
- *Data mining response time*. Time required by the Data mining module for performing the predictions at runtime. This time is subsumed by the Level-2 AM response time but still we find interesting to isolate it in our benchmarking.

Regarding the adequacy, two metrics are taken into account:

- *Adaptation enactment/decision correctness*. Expected adaptations, described in Table 29, that are finally realized.

– *Prediction correctness*. The need for (no) adaptation is correctly predicted and prediction results are timely, i.e., SSDV position after prediction is the same or previous to the last predicted.

Table 32 provides the replications' average response time (in milliseconds) for each use case scenario. We include the standard deviation of the response times. On the other hand, Figure 53 and Figure 54 present the detailed response time values obtained in each of the replications of each use case scenario. The x-axis of each sub-graph shows the number of replication, while the y-axis the response times. The Level-1 AM response time values obtained in the different scenarios are in average from 4,06 ms to 15,57 ms. While in us_2 this response time corresponds to the activation of a simulated V2V service; in **us4** and **us_5** the response time reflects a real service adaptation. In the implementation of HAFLoop for the SALI project, the deactivation of a service (**us_4**) consists in modifying a Boolean variable, while the frequency adaptation consists in killing an existing periodic process and creating a new one (**us_5**); this fact could explain the response time variation. Regarding Level-2 AM response time, values are in average from 291,06 ms to 892,18 ms. Comparing these results with the results obtained in SACRE (see Section 4.4.3), a great improvement can be noticed. In SACRE, resulting response time in scenarios using data mining was of the order of seconds, while in this evaluation results are of the order of milliseconds.

Table 32
Average response time per use case scenario

Use case scenario	Level-1 AM response time (ms)	Level-1 AM response time standard deviation (σ)	Level-2 AM response time (ms)	Level-2 AM response time standard deviation (σ)	Data mining response time (ms)	Data mining response time standard deviation (σ)
us1	N/A	N/A	855,34	23,34	113,93	13,31
us2	4,06	3,56	892,18	27,16	129,68	19,44
us3	N/A	N/A	875,35	18,35	130,47	16,39
us4	10,82	6,36	291,06	42,39	113,30	27,45
us5	15,57	8,62	297,00	51,50	115,03	30,45
us6	N/A	N/A	315,50	52,34	142,68	35,12

This improvement can be due to different factors. First, with the adoption of the HAFLoop4J framework, software modules now communicate to each other asynchronously. Second, normal operation and adaptation process are treated independently, i.e., modules are multi-thread. Third, the amount of data to analyze at runtime has been drastically reduced while ensuring its relevance, as suggested in the conclusions of SACRE. Finally, data mining is used for prediction and not for model generation. The significant difference in Level-2 AM's response time of sensor fault and battery issue scenarios is due to the waiting iterations of the Monitor and the Analyzer. That is, for **us_1-us_3,** the Monitor and the Analyzer wait 3 iterations before triggering an alert and the data mining process, respectively; meanwhile, for **us_4-us_6** the amount of iterations is 0.

Finally, regarding the Data mining module response time, it goes in average: from 113,30 ms to 142,68 ms. The Data mining module response time is composed of both the position and the self-

driving functionality usage predictions' response time. For **us1-us3**, this module's response time seems to be a small part of the Level-2 AM response time; this is because the total response time is affected by the waiting iterations mentioned before. The **us4-us6** scenarios reveal that the performance of this module does actually have a great impact; in these scenarios, it represents almost half of the Level-2 AM response time.

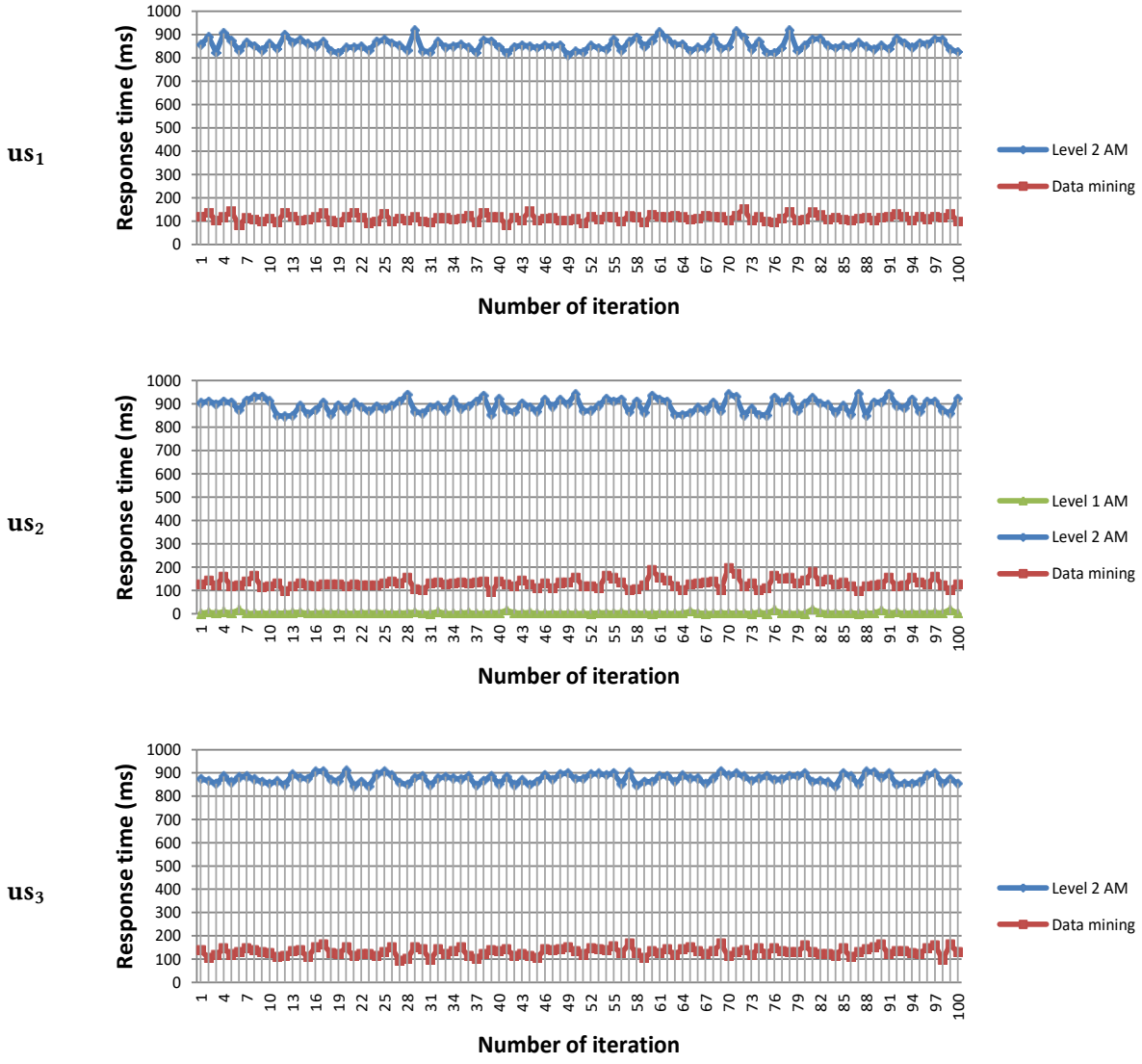


Figure 53: Adaptation response time per use case scenario replication (us_1-us_3)

In all the six use case scenarios, adaptation has been enacted when required and the decision of no adaptation needed has been correctly made (according to expected adaptations described in Table 25). Moreover, regarding the prediction correctness, both position and self-driving functionality usage have been predicted correctly in all the scenarios. Finally, in order to determine the prediction timeliness, we have plot for each scenario five metrics (see Figure 55):

- *Route*. The route followed by the SSDV during the execution of the scenario.
- *Self-driving active*. The segment in which the self-driving functionality is usually active, according to patterns learned (see Figure 52).
- *Sensor fault/battery issue position*. The average SSDV position, taking into account the results of all the replications at which the sensor fault or the battery issue is experienced.
- *Position after analysis*. The position of the SSDV after executing the data mining and providing the predictions.
- *Last predicted position*. According to the policies defined in Table 26, the Analyzer will try to predict the position of the SSDV in the following 100 iterations and the self-driving functionality usage in those positions. Therefore, the last predicted position corresponds to the 100th predicted.

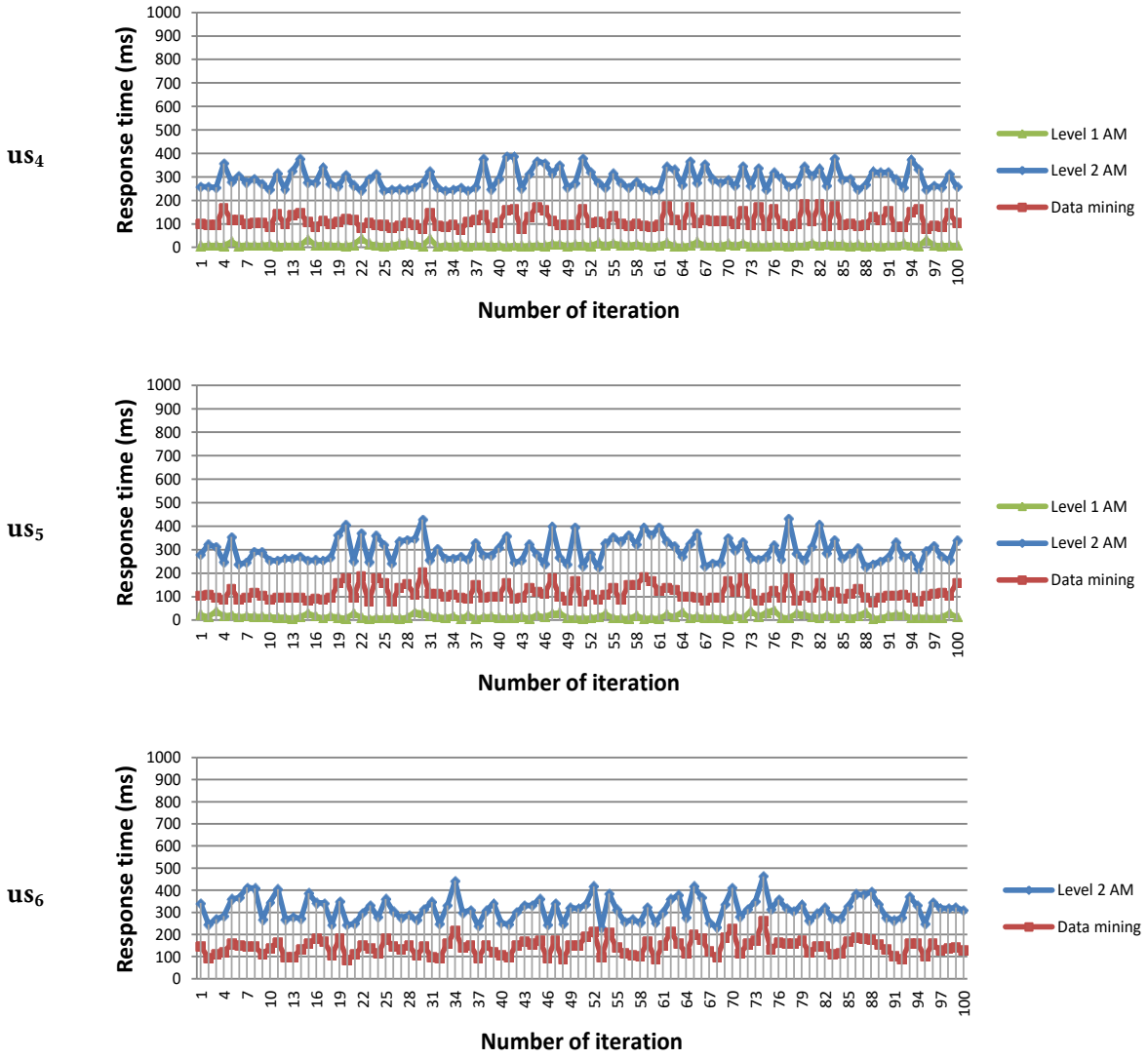


Figure 54: Adaptation response time per use case scenario replication (us_4-us_6)

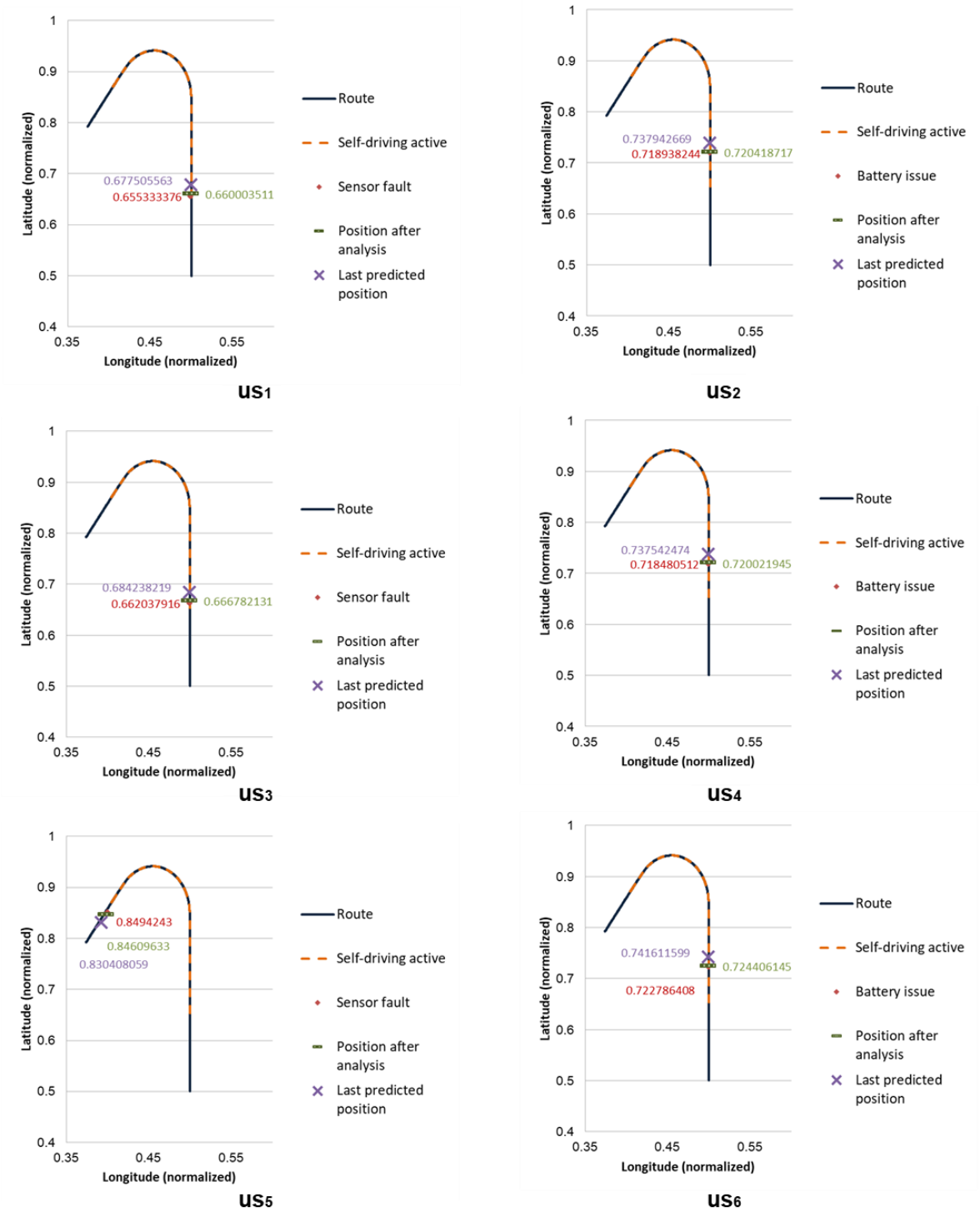


Figure 55: Prediction timeliness per use case scenario

According to the results shown in Figure 55, it can be concluded that in all the scenarios, the data mining predictions, apart from correct, have been timely performed. The number of positions to predict (100 in this evaluation) as well as the iterations to wait before triggering alerts and data

mining analysis (3 for the Monitor and Analyzer in this evaluation), as in SACRE, are exploratory variables and they may be further investigated. These parameters may depend on each application domain and sometimes even on each use case. The advantage of adopting HAFLoop and the HAFLoop4J framework is that SASs' owners can focus on investigating these kinds of domain-specific variables instead of spending resources on designing and constructing the generic functionalities of adaptive MAPE-K loops.

▸ Threats to validity

- *Internal validity*. The internal validity of this evaluation concerns to our ability to reason about the resulting self-improvement process' response time and adequacy, in each use case scenario, for instance, confounding variables' relationships. In order to reduce this threat, we have quantitatively interpreted our results using descriptive statistics for determine tendencies, dispersion and dependencies. Accidental bugs in software components are also a threat to internal validity. We have tried to reduce this unavoidable threat using well-established frameworks and tools for building our solution such as Spring boot (https://spring.io/projects/spring-boot), Gradle (https://gradle.org/), Docker, among others.
- *Construct validity*. In this evaluation, a threat to construct validity is that it was conducted using simulated components. Thus, the evaluation could be affected by our interpretation of the environment and the interactions of the driver with the SSDV. Moreover, factors that can only be measured in a real environment, e.g., time required by a sensor for physically turning on and off, could not be reflected in our evaluation results. In order to reduce this threat, we have utilized the OpenDaVINCI middleware, which offers a scaled environment with realistic road and vehicle dimensions, as well as real-time sensor data. OpenDaVINCI has been proposed as a standardized experimental platform for self-driving vehicles [125]. It enables efficient and riskless experiments and validation during the design process of solutions. Due to the standardized interfaces implemented in OpenDaVINCI, experiments' results can easily be transferred to real-scale vehicles.
- *External validity*. External validity refers to the generalizability of our conclusions. SALI evaluates HAFLoop in the domain of smart vehicles. The results have shown the feasibility and benefits of using our solution in this extremely demanding domain. However, due to the simulation environment in which the evaluation has been executed, generalization may be limited, not only to the domain, but also to the application of HAFLoop in this specific domain. This threat will be reduce with the evaluation of the SALI vehicle in a real environment. The details of this evaluation will be described in next section.

▸ Discussion

The SALI vehicle implements the fundamental ideas of our architectural proposal, HAFLoop, for correctly supporting adaptive monitoring in modern SASs. The implementation of SALI has satisfied the requirements identified in SACRE for better supporting the self-improvement process, i.e.,

support asynchronous communication, control the amount of data to analyze at runtime, ensure components reusability at a lower level, among others. The results of the evaluation of SALI in the simulation environment have not only confirmed the feasibility of adopting adaptive feedback loops in demanding application domains such the smart vehicles, but the benefits of supporting such feature. Thanks to the adoption of HAFLoop, challenging runtime factors affecting SSDVs, such as runtime unpredictable events and limited resources could be addressed. In next section, HAFLoop is incorporated in a real vehicle and the results of a series of experiments executed in a real scaled environment are presented.

4.6.3 SALI in a real environment

The self-driving vehicle

For conducting experiments in real vehicles, OpenDLV provides a series of low-level hardware-software interfaces for interacting with vehicles' sensors and actuators. OpenDLV has been developed in the context of the vehicle laboratory Chalmers Revere. This laboratory provides resources to researchers for the development and verification of software solutions in real vehicles. Using real vehicles provided by Revere (and AstaZero) and the facilities of the test ground AstaZero (http://www.astazero.com/), we were able to test HAFLoop in (controlled) real traffic environments. Concretely, we have utilized three Volvo cars: two XC90 and one V40. For collecting the evaluation data, we have enable self-improvement capabilities to one of the vehicles and utilize the other two for creating different scenarios. More details about the experiments conducted in AstaZero will be provided later in this section.

Most of the software modules running in the real vehicle are the same as the ones used in the simulated SDV (see Figure 45 and Figure 49). Policy variables were utilized for indicating the modules if the execution environment was simulation or real. In this section, we describe only the software modules that were not part of the simulated vehicle (see Figure 56):

- *Opendlv-device-gps*. This module interfaces with an Applanix POS GPS/INSS unit, providing data about the vehicle position (latitude and longitude, see Figure 57). More information about this module can be found at https://github.com/chalmers-revere/opendlv-device-gps-pos.
- *Opendlv-device-lidar*. This module interfaces with a VelodyneLidar HDL32e unit providing 360º 3D point cloud data (see Figure 58). LIDAR data is further processed by the extended version of OpenDLV[8] for reporting frontal, gear and lateral distances. Details about this component can be found at https://github.com/chalmers-revere/opendlv-device-lidar-hdl32e.
- *Opendlv-device-camera*. This module is utilized for interfacing with an Axis camera unit proving image data (see Figure 59). When receiving image data, a simulated image post-processing is performed, by the extended version of OpenDLV, and frontal distance data is

reported. More information about this component can be found at https://github.com/chalmers-revere/opendlv-device-camera-opencv.

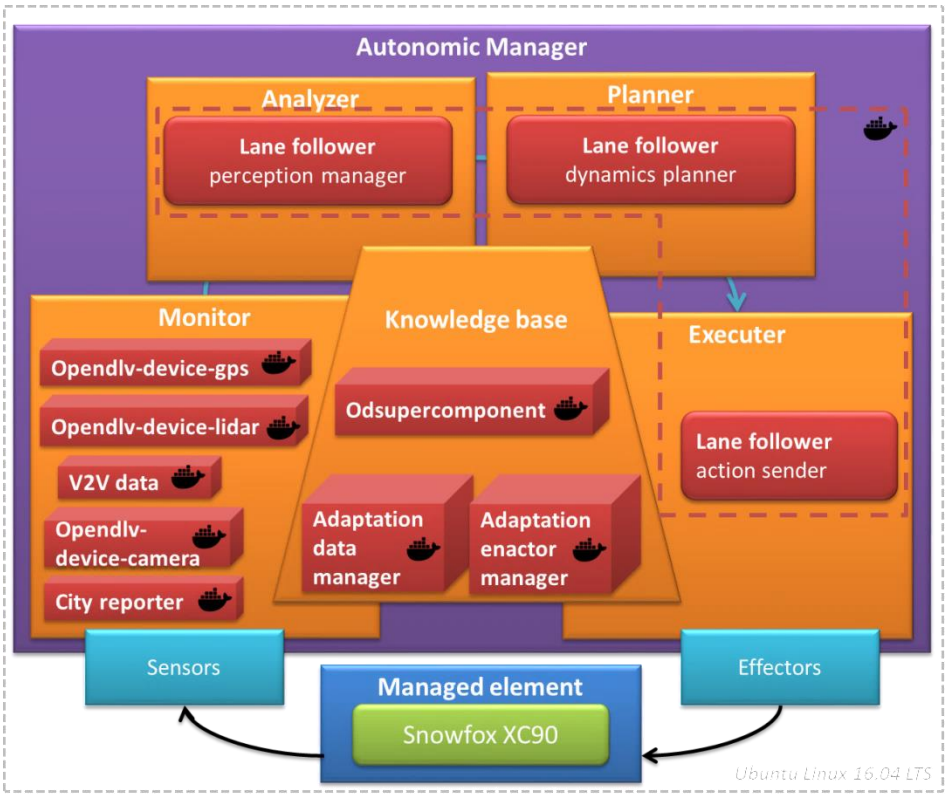


Figure 56: Self-driving vehicle (real environment)

Figure 57: SALI vehicle GPS data in the OpenDLV viewer

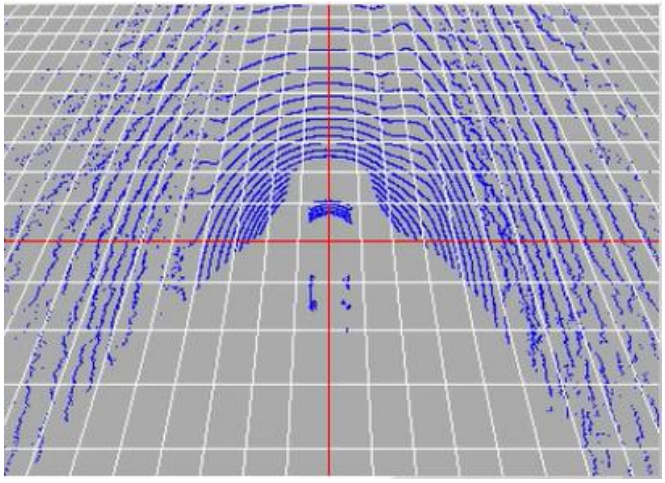


Figure 58: SALI vehicle lidar data (point cloud) in the OpenDLV viewer

Figure 59: SALI vehicle camera

In this implementation, the *lane follower* component is the same as in the simulated vehicle. The only difference is that in the real environment dynamics instructions are not executed by the real vehicle. Instead, a test driver simulates the functionality. For the self-improvement loop, i.e., the loop in charge of managing the adaptation of the Monitor element, the same implementation presented in Section 4.6.1 has been used in the real environment. Thus, in next section we present the evaluation of SALI.

4.6.4 Evaluation of SALI in a real environment

This evaluation aims at assessing in a real setting the performance of HAFLoop-based solutions for supporting adaptive monitoring in modern SASs, particularly in smart vehicles. Performance is evaluated in terms of both adaptation correctness and response time. The evaluation has been conducted at the AstaZero test track. We have run three use cases: the two use cases tested in the simulation environment, sensor fault and battery issues (see Section 4.6.2) and a use case of uncertainty, concretely, a road accident with uncertain sensor data. Experiments were conducted in two test areas: a scaled city area (see Figure 60) and a rural road (see Figure 61).

A soft vehicle (see yellow vehicle in Figure 62) and three real Volvo vehicles (see Figure 60 and Figure 61) were utilized for the experiments: two XC90 (Snowfox and Greyfox hereafter) and one V40. The Snowfox was selected for testing our proposal, while, the other two vehicles were utilized for creating the scenarios. Level-2 AM has been deployed on the same machine used for the simulation-based evaluation while Level-1 AM has been run on the vehicle's machine. Both machines have been connected through a local area network.

Figure 60: SALI project experiments execution at the AZ city area

Figure 61: SALI project experiments execution at the AZ rural road

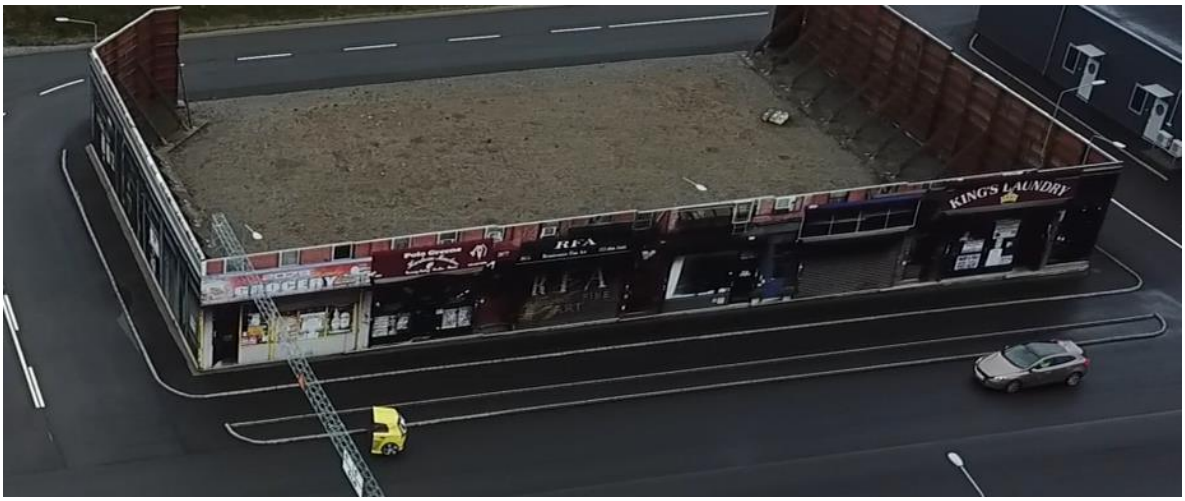


Figure 62: SALI project soft vehicle simulating a road accident at the AZ city area

Preparation activities

In order to evaluate SALI in the real environment, we have designed eight use case scenarios (us_1 to us_8 in Table 33). Figure 63-65 illustrate the different use cases. Similar to previous evaluation, a training phase has been conducted for predicting vehicles' position in the near future and the self-driving functionality usage. The road accident scenarios do not utilize the prediction feature. Therefore, the rural road has been utilized for performing the training phase. The driving scenario utilized exemplifies a driver that goes from work to home in a daily basis and utilizes the self-driving functionality in specific segments of that journey, as was the case if the simulation-based evaluation. The IBk [129] and JRip [38], [39] algorithms as well as the data mining Weka tool [40] have also

been used in the real environment. The resulting patterns of this training phase are illustrated in Figure 66.

Table 33
Use case scenarios

Id	Challenging factor	Scenario	Expected adaptation
us_1	Road accident	A road accident notification is received and traffic data gathered by the city reporter service indicates normal traffic load, and parameter adaptation is enabled.	Traffic monitoring frequency is increased in order to gather fresher data.
us_2		A road accident notification is received and traffic data gathered by the city reporter service indicates normal traffic load, and parameter adaptation is disabled.	Traffic monitoring is deactivated and V2V communication is utilized instead in order to improve traffic data accuracy (1st adaptation). After route re-calculation and change, city reporter traffic monitoring is re-activated and V2V deactivated (2nd adaptation).
us_3	Sensor fault	LIDAR sensor fails when the SSDV goes on a road with no other vehicles, at the beginning of the journey.	Axis camera is activated.
us_4		LIDAR sensor fails when the SSDV goes on a road with other vehicles, at the beginning of the journey.	Axis camera plus V2V communication is activated given the increased driving risk. Self-driving functionality stays active.
us_5		LIDAR sensor fails when the SSDV goes on a road with no other vehicles, close to the end of the journey.	No monitor adaptation is enacted. According to patterns, driver will change to manual mode in the near future.
us_6	Battery issues	Critical battery level is experienced when the SSDV starts its journey in a road with no other vehicles, and parameter adaptation is disabled.	A trade-off between required and non-required monitor is performed, resulting in the deactivation of the city reporter and the axis camera.
us_7		Critical battery level is experienced when the SSDV starts its journey in a road with other vehicles, and parameter adaptation is disabled.	No monitor adaptation is enacted given the increased driving risk. However, a take-over request is sent to the driver. Driver mode is changed to manual
us_8		Critical battery level is experienced when the SSDV goes on a road with no other vehicles, close to the end of the journey.	No monitor adaptation is enacted. According to patterns, driver will change to manual mode in the near future.

Figure 63: Road accident use case

Figure 64: Sensor fault use case

Figure 65: Battery issues use case

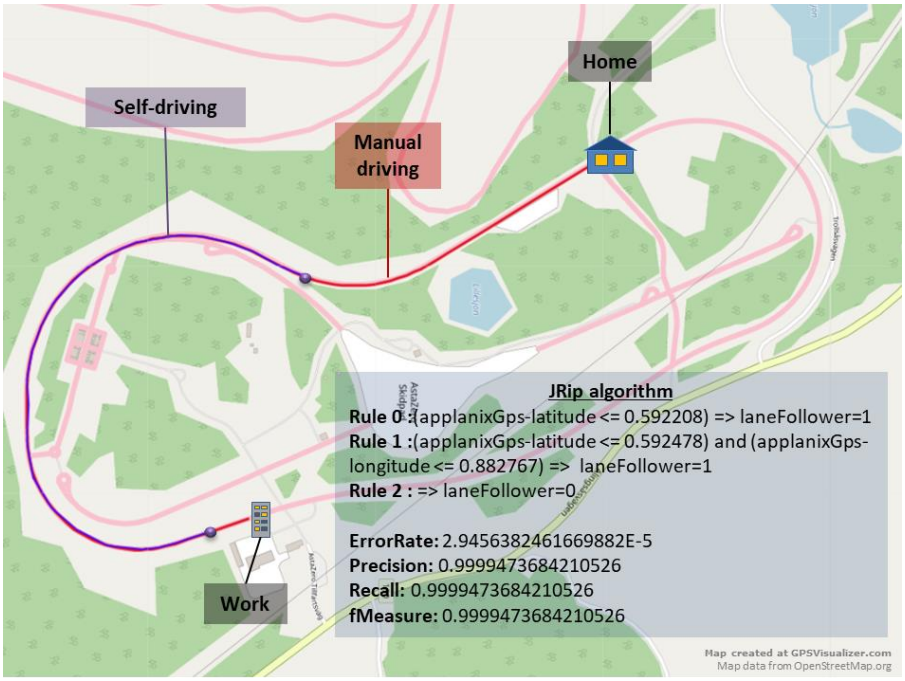


Figure 66: SALI learning phase patterns (real environment)

Regarding policies, Table 34 and Table 35 provide the policy variables' values we have set for the Level-2 and Level-1 AM, respectively.

Table 34
Level-2 AM policies

Policy element	Variable	Variable description	us1	us2	us3	us4	us5	us6	us7	us8
Monitor	Alert iterations	Number of iterations, detecting sensor fault or battery issue, to wait before triggering an analysis alert	0	1st adapt: 0 2nd adapt: 3	3			0		
	Initial battery level	The battery level at the initial point of each scenario execution	100%					50%		60%
	Battery limit	The battery level considered as critical	40%							
	Monitors	List of monitoring data sources, type of source (T), the monitoring data provided (Vars), the monitoring frequency (F) and their cost (C, a factor in relation to the rest of monitors, taking into account power and monetary aspects, and its utility for correctly supporting the self-driving functionality)	**traffic**: (T) service, (Vars) traffic factor, (F) 60000ms, (C) 6.7 **weather**: (T) service, (Vars) weather, (F) 60000 ms, (C) 6.7 **can**: (T) sensor, (Vars) speed, (F) 10 ms, (C) 0.1 **camera**: (T) sensor, (Vars) image size – frontal distance, (F) 50 ms, (C) 3.5 **gps**: (T) sensor, (Vars) longitude – latitude, (F) 100 ms, (C) 28.5 **lidar**: (T) sensor, (Vars) end and start azimuth – frontal, right, left and rear distance, (F) 100 ms, (C) 2.7 **V2V**: (T) service, (Vars) traffic factor – frontal distance – road event, (F) 100 ms, (C) 40							
	Monitoring variables	Variables to be monitored and variables' values characteristics (type (T), min, max or possible values (Val)). Values out of min-max range or not listed as possible values, might indicate a monitor fault	**traffic factor**: (T) Double, (Val) -1, 10 **weather**: (T) String, (Val) Rain, Snow, Extreme, Clear, Clouds, Foggy, Fog, Drizzle, Mist **image size**: (T) Double, (Val) 0, 5000000 **longitude**: (T) Double, (Val) 11.0, 13.0 **latitude**: (T) Double, (Val) 56.0, 59.0 **speed**: (T) Double, (Val) 0, 120 **start azimuth**: (T) Double, (Val) 0, 1 **end azimuth**: (T) Double, (Val) 358, 360 **frontal right distance, rear right distance, frontal center distance, rear distance**: (T) Double, (Val) -1, 10000 [cm] **road event**: (T) String, (Val) Crash							
	Initial monitors	Initial set of active monitoring data	can, lidar, gps, traffic, weather,		can, lidar, gps, traffic, weather,			can, lidar, gps, traffic, weather,		

		sources	camera			camera
Analyzer	Alert iterations	Number of iterations, receiving an alert from the Monitor, to wait before triggering Data mining analysis	0	1st adapt: 0 2nd adapt: 3	3	0
	Adaptation supported	Type(s) of adaptation supported	Para.	Structure		
	Analysis technique	This could include: technique, tool, endpoint, algorithms, algorithms' parameters	**Technique**: Machine Learning **Tool**: Weka **Endpoint**: protocol, host, port **Algorithms**: JRip, IBk **Parameters**: Positions to predict (N = 500)			
	ME functiona-lities	Critical functionalities to be provided by the ME	Self-driving			
Planner	Plan technique	This could include: technique, tool, endpoint, algorithms, algorithms' parameters	Technique: Objective function (min (monitoring cost), max (monitoring data required by the ME functionalities))			
	Monitoring data required by ME functiona-lities	Monitoring data required by each of the ME functionalities	Self-driving: frontal, right, left and rear distance, longitude, latitude, speed			
Executer	Level-1 AM endpoint	ME interface to communicate adaptation decisions	protocol, host, port			
Know-ledge base	Persistence format	The format in which data is going to be persisted	.arff (the format required by the Weka tool)			
	Monitoring data	List of monitoring data sources to take into account for persistence	traffic, can, camera, weather, lidar, gps, V2V			

▶ Scenarios execution and analysis of results

The execution of the experiments was done in the context of the SALI project. Six testing days were utilized for conducting the experiments, two days per use case. The execution of the experiments has consisted on running the different scenarios, repeatedly. In total, 30 executions have been computed. In order to analyze the evaluation results, two aspects of the self-improvement process are explored: response time and adequacy. The response time is split into three categories.

Table 35
Level-1 AM policies

Policy element	*Variable*	*Variable description*	us_1	us_2	us_3	us_4	us_5	us_6	us_7	us_8
Monitor	Monitors	List of monitoring data sources and their monitoring frequency	**traffic**: (T) service, (Vars) traffic factor, (F) 60000ms, (C) 6.7 **weather**: (T) service, (Vars) weather, (F) 60000 ms, (C) 6.7 **can**: (T) sensor, (Vars) speed, (F) 10 ms, (C) 0.1 **camera**: (T) sensor, (Vars) image size – frontal distance, (F) 50 ms, (C) 3.5 **gps**: (T) sensor, (Vars) longitude – latitude, (F) 100 ms, (C) 28.5 **lidar**: (T) sensor, (Vars) end and start azimuth – frontal, right, left and rear distance, (F) 100 ms, (C) 2.7 **V2V**: (T) service, (Vars) traffic factor – frontal distance – road event, (F) 100 ms, (C) 40							
	Monitoring variables	Variables to be monitored	**traffic factor**: (T) Double, (Val) -1, 10 **weather**: (T) String, (Val) Rain, Snow, Extreme, Clear, Clouds, Foggy, Fog, Drizzle, Mist **image size**: (T) Double, (Val) 0, 5000000 **longitude**: (T) Double, (Val) 11.0, 13.0 **latitude**: (T) Double, (Val) 56.0, 59.0 **speed**: (T) Double, (Val) 0, 120 **start azimuth**: (T) Double, (Val) 0, 1 **end azimuth**: (T) Double, (Val) 358, 360 **frontal right distance, rear right distance, frontal center distance, rear distance**: (T) Double, (Val) -1, 10000 [cm] **road event**: (T) String, (Val) Crash							
	Initial monitors	Initial set of active monitoring data sources	can, lidar, gps, traffic, weather, camera		can, lidar, gps, traffic, weather			can, lidar, gps, traffic, weather, camera		
Analyzer	Vehicle variables	Variables resulting from vehicle's monitoring	vehicle's position and speed, frontal, rear, right, and left distance							
	Context variables	Variables gathered from external systems	weather, traffic, road event							
Planner	Adaptation variables	Variables to include in the adaptation plan	Acceleration, steering wheel angle, driving route							
Executer	ME endpoint	ME interface to communicate adaptation decisions	vehicle id							
Knowledge base	Level-2 AM sensors endpoint	Interface to send runtime data to Level-2 AM	protocol, host, port+							

- *Level-2 AM response time.* Time elapsed since the Monitor element detects a challenging factor (from Table 33) until a decision of no adaptation required or an adaptation request is sent to the Level-1 AM.
- *Level-1 AM response time.* Time elapsed since an adaptation request is received until it is enacted.
- *Data mining response time.* Time required by the data mining module for performing the predictions at runtime, i.e. the prediction of the vehicle's position in the near future and the prediction of the self-driving functionality usage in that position. This time is subsumed by the Level-2 AM response time but still we find interesting to isolate it in our benchmarking for comparing it with previous evaluations.

For the self-improvement adequacy, we have first evaluated the correctness of the adaptation enactment process, i.e., we have determined whether the expected adaptations, described in Table 33, are finally enacted and checked if they are enacted in a timely way (from a human-perspective). After the execution of all the use cases scenarios, we have been able to confirm that all the adaptation decisions have been executed correctly and as expected. From a human-perspective, we have also determined that they have been timely enacted as follows. For the road accident use case, the adaptation has been executed before the vehicle reaches the intersection (see Figure 63); therefore, the self-driving logic has been able to re-calculate the route to home, on time. For the sensor fault use case, when an adaptation has been enacted, it was performed within the road segment in which the self-driving functionality is typically used (according to patterns on Figure 66). Similarly, it has happened for the battery issues use case, i.e., sensors trade-off has also been performed within the segment where self-driving was required.

We have also evaluated the self-improvement adequacy in terms of the prediction correctness. For all the use cases, the prediction on the self-driving usage (and in consequence, the prediction of the vehicle's position) has been accurate and timely. That is, when the vehicle was at the beginning of the journey the prediction has indicated that functionality will be used while when it was at the end of the segment where self-driving is usually used (see Figure 66), the prediction was the other way around. This has allowed the vehicle to correctly made the decisions of (no) adaptation described before. Table 36 provides the average response time (in milliseconds) of the self-improvement loop (Level-2 AM). Response time of use cases requiring data mining is differentiated from the one that does not. We have also reported separately the response time of the scenarios that considered a minimum number of Monitor and Analyze alert iterations from the ones that do not. Table 37 provides the resulting average response time of the Data mining module, i.e., the predictions. We have included the standard deviation of the different response times in both tables. In Figure 67, the detailed response time values obtained in each use case scenario execution are provided. The x-axis shows the number of execution, while the y-axis the response times.

Table 36
Self-improvement loop response time

Involved use case scenarios	Challenging factors	Monitor/ Analysis alert iterations	Self-improvement loop avg. response time (ms)		Self-improvement loop response time std. dev.	
			With data mining	*Without data mining*	*With data mining*	*Without data mining*
us_1-us_2	Road accident	0	N/A	249,83	N/A	232,27
		3	N/A	781,5	N/A	192,61
us_3-us_5	Sensor fault	3	4421,5	N/A	915,14	N/A
us_6-us_8	Battery issues	0	3971,75	N/A	739,49	N/A

Table 37
Data mining response time

Involved use case scenarios	Challenging factors	Data mining avg. response time (ms)	Data mining response time std. dev.
us_1-us_2	Road accident	N/A	N/A
us_3-us_5	Sensor fault	*3624*	*838,52*
us_6-us_8	Battery issues		

The resulting Level-2 AM response times go in average from 249,83 ms to 781,5 ms when data mining is not required and from 3971,75 ms to 4421,5 ms when it is required. For the second case, in average 3624 ms are spent by the Data mining module. In both case, there is an increment of the response time (around 500 ms) for the scenarios considering waiting alert iterations, e.g., the second adaptation of uc_2. As in previous evaluations, the waiting alert iterations are exploratory variables that may be further investigated; it may depend on each application domain or even on each use case. Regarding the adaptations' enactment, for uc_2 and uc_6 adaptation involved changes of OpenDLV components (Camera and V2V communication) as well as of the City reporter service (Traffic monitoring). In these cases, adaptation requests have been sent sequentially, i.e., first to the cloud service and then to the sensors. This can be noticed in Figure 67, where Level-2 AM response time for the City reporter is clearly smaller than for OpenDLV. This issue depends directly on the implementation and it can be easily fixed with parallelism. For the purposes of this evaluation, we have averaged both response times.

Comparing our results to the experiments executed in the simulation environments, a great improvement regarding SACRE can be noticed. In SACRE, response times went in average from 260

ms to 2470 ms when data mining was not required and from 3850 ms to 30260 ms when it was. This improvement can be due to the different factors mentioned in the analysis of the results of the evaluation presented in Section 4.6.2. Regarding the last evaluation performed in a simulation environment (see Section 4.6.2), Level-2 AM response times are greater. Considering that most of the software modules have been reutilized for the evaluation presented in this section, one explanation of the increased response times can be the complexity of the execution contexts. For instance, due to the real vehicle speed (around 20 km/h for us_1-us_2 and 40-50 km/h for the rest of scenarios) the amount of predicted time steps (positions) has been increased to 500 (5 times the valued used in the evaluation of Section 4.6.2). Although, the response time is greater, it is acceptable for a real environment. Further research may investigate optimization methods for improving our results.

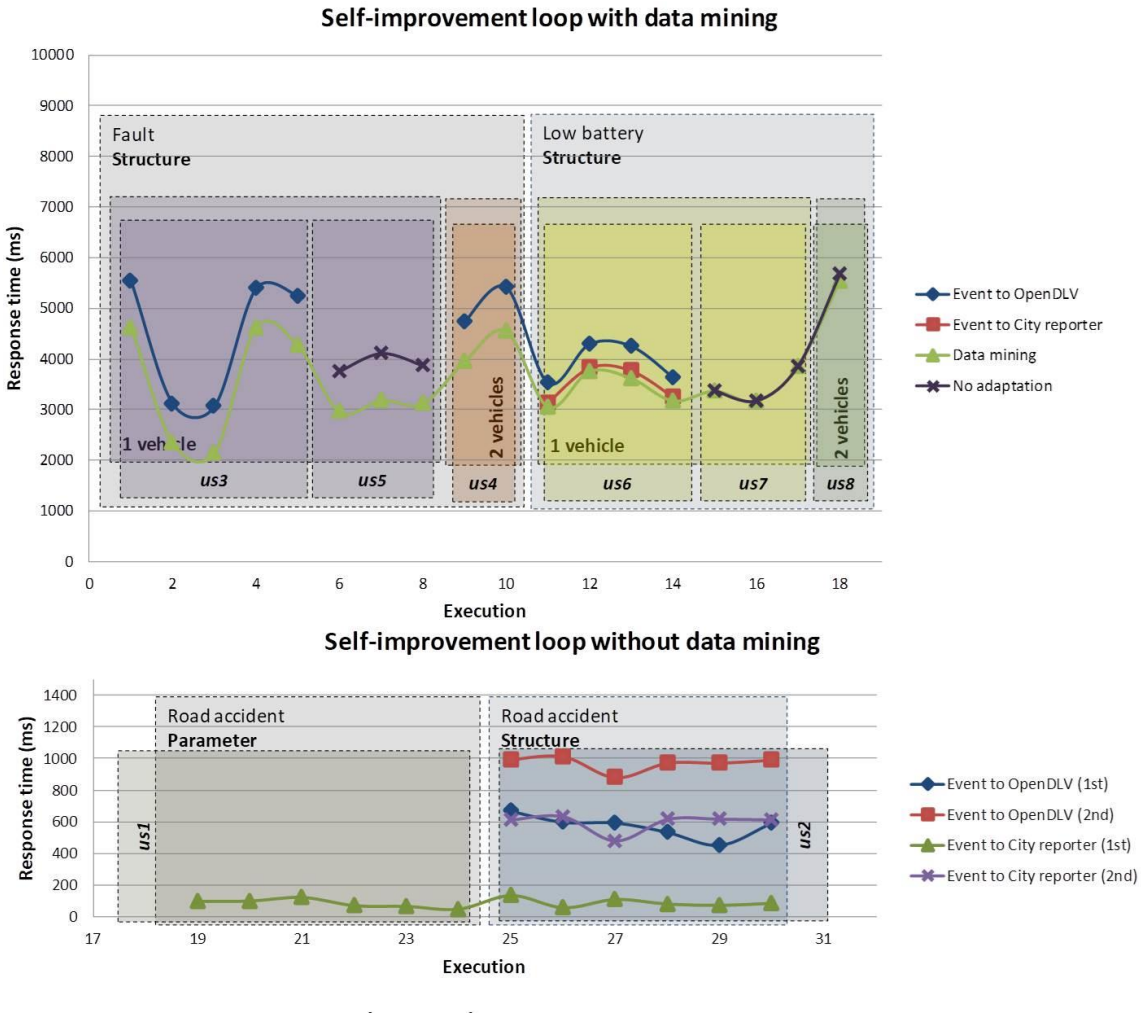


Figure 67: Self-improvement loop response time per execution

Table 37 provides the average response time (in milliseconds) of the Adaptation loop (Level-1 AM). Response times are reported by each monitoring system adapted, i.e., Camera, V2V or City reporter service. For the Camera and V2V communication, the de/activation has been simulated by software

components; thus, the enactment response time in real cases may differ from what is reported in this evaluation. On the order hand, the response time of the City reporter does actually reflect a real service adaptation, although services implementations may vary in several ways. We have included the standard deviation of the different response times. In Figure 68, the detailed enactment response times obtained in each use case scenario execution are provided. Each monitoring system's enactment time is provided in a different graph. The x-axis of the graphs show the number of execution, while the y-axis the response times.

Table 38
Adaptation loop response time

Involved use case scenarios	Challenging factors	Adaptation loop average response time (ms)			Self-improvement loop response time std. dev.		
		Camera	*V2V*	*City reporter*	*Camera*	*V2V*	*City reporter*
us_1-us_2	Road accident	N/A	9,5	298,95	N/A	8,24	176,92
us_3-us_5	Sensor fault	1,91			4,5		
us_3-us_5	Battery issues		N/A			N/A	

Focusing on the results of the City reporter service, one can notice that for the second adaptation of us2, the response time is much smaller than for the first adaptation (almost of the order of 20). This can be due to the service communication protocol, which could have affected the first communication established between both services. However, no conclusions can be done since an existing framework for managing such communication has been used (i.e., Spring boot); therefore, low-level details are hidden from our perspective. What can be concluded is that this phenomenon is systematic, as it was present in all the executions; therefore, an explanation should be possible. More experiments could be executed for exploring this factor, for instance involving a 3rd (or more) adaptation (s).

The evaluation of HAFLoop in a real environment has provided promising results regarding the feasibility and benefits of supporting adaptive monitoring in modern SASs such the SDVs. Our solution has enabled a SDV to respond to runtime challenging events such as uncertain sensor data, sensor faults, and battery issues, accurately and timely. Among the benefits of using HAFLoop, it can be remarked: the ability of SASs to implement self-improvement capabilities and in this way adapt their feedback loops at runtime for better supporting MEs. Without adopting a self-improvement approach such HAFLoop, this would not be possible. Moreover, given the generalizability level of HAFLoop (and its implementation, HAFLoop4J), it is easy to adopt, reuse and extend for supporting a variety of SASs as well as different types of adaptation, addressing the principle challenges affecting this field.

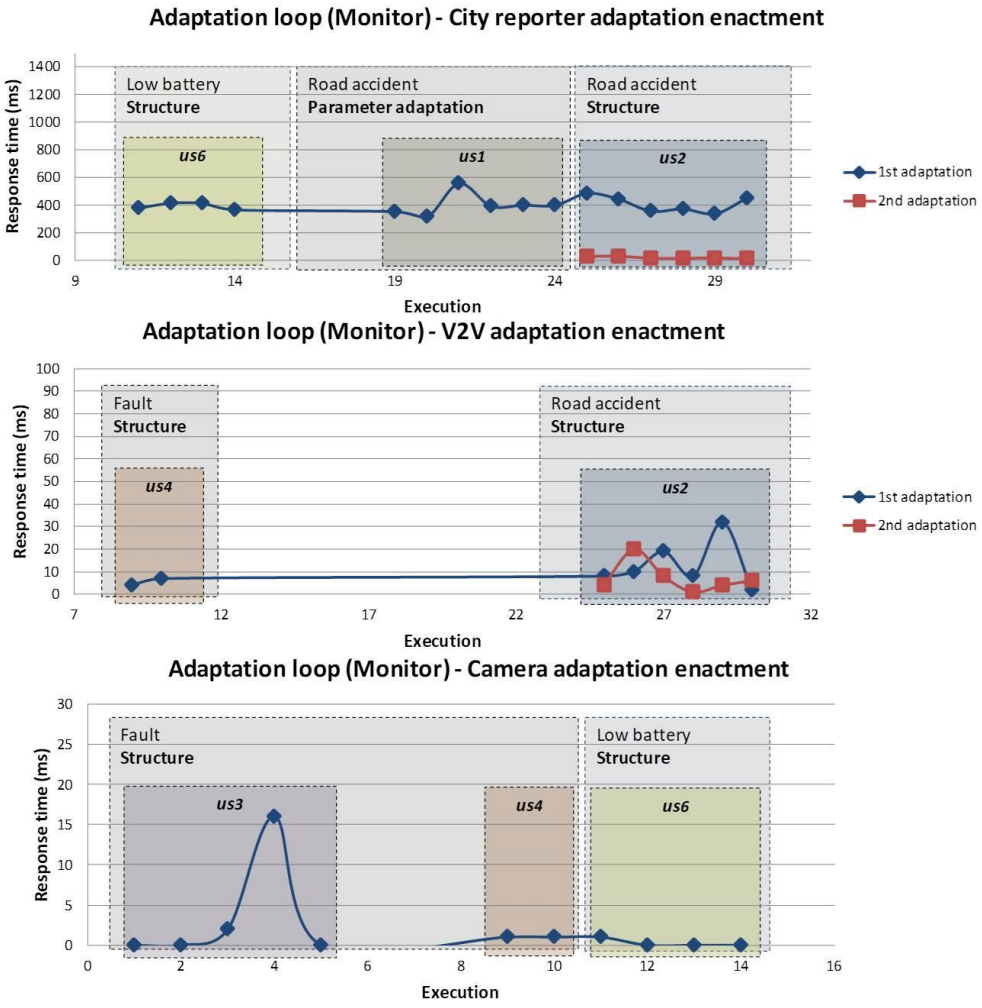


Figure 68: Adaptation loop response time per execution

Threats to validity

- *Internal validity*. The internal validity of this evaluation concerns to our ability to reason about the resulting self-improvement process' response time and adequacy, in each use case scenario, for instance, confounding variables' relationships. In order to reduce this threat, we have quantitatively interpreted our results using descriptive statistics for determine tendencies and dispersion. Accidental bugs in software components are also a threat to internal validity. We have tried to reduce this unavoidable threat using well-established frameworks and tools for building our solution such as Spring boot, Gradle, Docker, among others.
- *Construct validity*. In this evaluation, a threat to construct validity is that some of the vehicle's dynamics and sensors' adaptation have been simulated. Therefore, some factors cannot be measure, e.g., time required by a sensor for physically turning on and offs. In order to reduce this threat, we have incorporated a monitoring service that supports both structural and parameter adaptation and that reflects a more realistic runtime adaptation response time.

- *External validity*. External validity refers to the generalizability of our conclusions. This work evaluates HAFLoop in the domain of smart vehicles. The results show the feasibility and benefits of using our solution in this extremely demanding domain. However, the evaluation scenarios are simple compared to the complex situations an AV may experience in real life; therefore, generalization of the results without taking into account other factors may be limited, not only to the domain, but also to the application of HAFLoop in this specific domain. Although more experimentation may reduce this threat, given the great diversity of SASs' execution contexts, it will always exist.

▶ Discussion

From the results analyzed in this thesis, and our experience at AstaZero, we can say that HAFLoop is a promising proposal for supporting adaptive monitoring in modern SASs such the SDVs. Moreover, it also demonstrates the feasibility of integrating data mining techniques in complex SASs at runtime. From the SDVs perspective, our solution enables systems to respond to runtime challenging events affecting this application domain, both accurately and timely. The adoption of the HAFLoop4J framework has fastened the development of the solution as well as its portability from one environment to another, e.g., using policies for indicating the evaluation environment. Thanks to the modularity of the software components at low-level, further experiments with different considerations can be easily executed for evaluating the adoption of self-improvement capabilities in existing SASs, particularly in SDVs. For example, the data mining algorithms can be replaced, sophisticated decision-making techniques can be integrated, or the waiting alert iterations can be adjusted.

V

Conclusions and future work

In this thesis, we have addressed the automatic runtime adaptation of SASs' AM, particularly the Monitor element, in order to respond to changes in the MEs, the environment and the AM itself. Concretely, we have presented HAFLoop, an architectural proposal for supporting the SASs' self-improvement property. We have identified open research challenges affecting SASs' and AMs' adaptation at runtime and analyzed whether and how existing approaches address those challenges. We have also studied how state-of-the-art approaches support adaptive monitoring in current monitoring systems. Although great efforts have been done for supporting the adaptation of SASs' AM, none of the state-of-the-art solutions satisfactorily addresses all the open research challenges. Motivated by this fact, we have developed HAFLoop. HAFLoop, in conjunction with its implementation HAFLoop4J, is a generic and reusable solution that easies the design and development of adaptive AMs in modern SASs, from higher to lower levels. Our solution enables AMs to support different types of adaptation in a variety of settings. HAFLoop has been evaluated in different scenarios and in both simulation and real environments. The evaluations of HAFLoop have been conducted in the domain of smart vehicles with very promising results. In the introduction of this thesis, we have stated three RQs. In the rest of this section, we provide answers to these RQs and discuss the possible future work from the current state of the research.

5.1 Conclusions of RQ1

In order to address **RQ1**, we have conducted a systematic mapping study on adaptive monitoring focused on the adaptation of the elements directly related to the data gathering activity. The study aims at giving a comprehensive overview of the current state of the art of the adaptive monitoring topic and improving the understanding about how approaches from different research fields (tend to) conduct the adaptation process. For this purpose, we have followed a systematic review protocol that has allowed us to identify 110 studies organized in 81 proposals for supporting adaptive monitoring in a variety of research fields. The studies have been used for addressing a series of research questions we have defined as part of the review process. The analysis has been thorough, relying on coding and Data Mining for a deep understanding of the answers to the research questions.

We consider that the results we have obtained can be useful in the standardization of adaptive monitoring concepts (e.g., utilizing the codes we have developed for describing the different elements), as well as in the development of more complete, flexible, reusable and generic software engineering solutions for supporting adaptive monitoring in a variety of systems. From our side, we have proposed a generic definition for the term *adaptive monitoring*, based on our findings in the SMS. Moreover, in this thesis we propose a software engineering solution that satisfies the requirements of modern systems such are the SASs. Our solution can support any type of monitor adaptation and provides a reusable architecture that coordinates normal monitors' operation with their adaptation process. In order to do that, our solution for adaptive monitoring separates generic from system-specific functionalities. Moreover, we have developed a framework that easies the systematic development of adaptive monitors.

5.2 Conclusions of RQ2

In order to address **RQ2**, we have conducted two literature reviews, one driven by current requirements and engineering challenges affecting SASs, and a second one on the driven by the current challenges affecting SASs' self-improvement. The reviews aim at providing an overview of the current state-of-the-art approaches dealing with the adaptation of the AM of SASs at runtime. As well as uncovering the research gaps of this topic. For this purpose, we have followed a systematic review protocol that has allowed us to identify: in the first case, four approaches for supporting the adaptation SASs' capabilities (i.e., adaptation requirements): in the second case, 25 articles organized in 17 proposals for supporting SASs' self-improvement. The approaches' proposal have been characterized and analyzed in terms of whether and how they address current research challenges affecting SASs.

We consider that the results we have obtained can be useful in the understanding of the current research challenges that affect the field of SASs in general and the support of the self-improvement property in particular. Moreover, our results may motivate new proposals

towards more flexible, in terms of (de)centralization levels, more complete, in terms of supporting different types of adaptations, and more reusable and generic, in terms of supporting the software engineering life cycle from higher to lower implementation levels of this kind of systems. From our side, in this thesis we propose a software engineering solution that addresses some of the most relevant challenges of modern SASs. Our solution not only can support the adaptation of the Monitor element of SASs' AM, but a complete adaptive AM. Moreover, thanks to its generalizability level, any the different types of adaptation described in challenges can be supported. In the form of a framework, we offer extensible and reusable software components that easy the implementation of adaptive MAPE-K loops for SASs. Using a set of policies implementations can be customized and applied in a variety of domains.

5.3 Conclusions of RQ3

In order to address **RQ3**, we have presented HAFLoop, a generic and highly-modular SASs' self-improvement architecture able to support the adaptive AMs in modern SASs, where decentralization and cooperation are highly important characteristics. In **RQ1** and **RQ2**, we have identified the requirements and open research challenges affecting adaptive monitoring systems and SASs' self-improvement, respectively. Through systematic literature reviews, we have shown current solutions for supporting the adaptation of monitoring systems and the adaptation of SASs' AM, however, none of them satisfactorily addressed all the current challenges. Motivated by this fact, we proposed HAFLoop. The main features of HAFLoop have been designed based on the current needs of SASs. That is:

- Generic and reusable approaches
- Support both reactive and proactive adaptation
- Include structural adaptation of the AM
- Support adaptation on different settings, from centralized to fully decentralized

We have implemented HAFLoop in the form of a framework of Java-based applications, ensuring the satisfaction of SASs' needs even at the low implementation level. Asynchronous communication mechanisms and separation of concerns at the components' knowledge level, allows HAFLoop to support a variety of settings. Moreover, HAFLoop can be, fully or partially implemented, not only in terms of AM elements' adaptation capabilities but also in terms of separated MAPE-K elements. Thus, for instance, an adaptive monitoring system could be implemented in isolation for a different purpose that does not involve SASs. However, that is out of the scope of this thesis.

HAFLoop has been validated in a variety of use cases scenarios and in both simulation and real environments. The evaluation has been conducted in the extremely demanding domain of smart vehicles, where both the correctness of the functionality and the response time are very important factors. Our proposal has demonstrated to be not only suitable for such kind of systems but also useful for dealing with challenges affecting that specific application domain.

5.4 Future work

This work can be extended in different directions:

1) The SMS study on adaptive monitoring can be extended to answer new research questions of interest for the community. For instance, to assess the specific techniques utilized for analyzing runtime data and the decision-making approaches.
2) In the SMS, we have introduced the use of Data Mining for analyzing approaches solutions and finding hidden patterns among them. The analysis can be extended, comparing different algorithms and techniques for performing this task.
3) Regarding the literature reviews on SASs' self-improvement, it could be extended to answer other research questions such as which are the terms utilized by the community for referring to the adaptation of the AM. For instance, one of our findings was that researchers also utilized the word evolution for referring to the same process.
4) A next step for HAFLoop would be to developed generic approaches for the adaptation of each of the MAPE-K elements. Given their differences in nature, specific approaches may be studied for each of the elements. For instance, the adaptation of the frequency parameter may be interesting for the Monitor element, but no relevant for the Planner.
5) Finally, regarding the evaluation of HAFLoop, it can be extended with experiments in other application domains as well as by the utilization of different techniques in the domain of smart vehicles, e.g., using different analysis techniques or different algorithms.

VI

Bibliography

[1] R. De Lemos, H. Giese, H. A. Müller, and M. Shaw, "Software engineering for self-adaptive systems: A second research roadmap," *Softw. Eng. Self-Adaptive Syst. II*, vol. 7475, 2013.

[2] C. Krupitzer, F. M. Roth, S. Vansyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervasive Mob. Comput.*, vol. 17, no. PB, pp. 184–206, Feb. 2015.

[3] B. Cheng *et al.*, *D*, vol. 5525. eyns, and J. Whittle. Software Engineering for Self-Adaptive Systems: A Research Roadmap. Springer Berlin Heidelberg, Lecture Notes in Computer Science, 2009.

[4] D. Weyns *et al.*, "On patterns for decentralized control in self-adaptive systems," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7475 LNCS, pp. 76–107, 2013.

[5] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *IEEE Comput. Soc.*, vol. 36, no. 1, pp. 41–50, 2003.

[6] IBM-Corporation, "An architectural blueprint for autonomic computing," *IBM White Pap.*, vol. 36, no. June, p. 34, 2006.

[7] E. Zavala, X. Franch, J. Marco, A. Knauss, and D. Damian, "SACRE: Supporting contextual requirements' adaptation in modern self-adaptive systems in the presence of uncertainty at runtime," *Expert Syst. Appl.*, vol. 98, pp. 166–188, May 2018.

[8] A. Toueir, J. Broisin, and M. Sibilla, "A goal-oriented approach for adaptive SLA monitoring: A cloud provider case study," in *2nd IEEE Latin American Conference on Cloud Computing and Communications (LatinCloud)*, 2013, pp. 53–58.

[9] A. J. Ramirez, B. H. C. Cheng, and P. K. McKinley, "Adaptive monitoring of software requirements," in *1st International Workshop on Requirements@Run.Time (RE@RunTime)*, 2010, pp. 41–50.

[10] A. Moui and T. Desprats, "Towards self-adaptive monitoring framework for integrated management," in *IFIP International Conference on Autonomous Infrastructure, Management and Security (AIMS)*, 2011, vol. 6734 LNCS, pp. 160–163.

[11] G. Tamura, N. M. Villegas, H. A. Muller, L. Duchien, and L. Seinturier, "Improving context-awareness in self-adaptation using the DYNAMICO reference model," in *2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2013, pp. 153–162.

[12] T. Zhao, W. Zhang, H. Zhao, and Z. Jin, "A Reinforcement Learning-Based Framework for the Generation and Evolution of Adaptation Rules," in *Proceedings - 2017 IEEE International Conference on Autonomic Computing, ICAC 2017*, 2017, pp. 103–112.

[13] M. U. Iftikhar and D. Weyns, "Assuring system goals under uncertainty with active formal models of self-adaptation," in *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*, 2014, no. 1, pp. 604–605.

[14] R. J. Anthony, M. Pelc, and W. Byrski, "Context-aware Reconfiguration of Autonomic Managers in Real-time Control Applications," in *Proceeding of the 7th international conference on Autonomic computing - ICAC '10*, 2010, pp. 73–74.

[15] C. Krupitzer, J. Otto, F. M. Roth, A. Frommgen, and C. Becker, "Adding Self-Improvement to an Autonomic Traffic Management System," in *Proceedings - 2017 IEEE International Conference on Autonomic Computing, ICAC 2017*, 2017, pp. 209–214.

[16] C. Krupitzer, F. M. Roth, M. Pfannemuller, and C. Becker, "Comparison of approaches for self-improvement in self-adaptive systems," in *Proceedings - 2016 IEEE International Conference on Autonomic Computing, ICAC 2016*, 2016, pp. 308–314.

[17] M. Shaw, "Coming of Age of Software Architecture Research," in *23rd IEEE International Conference on Software Engineering*, 2001, pp. 656–664.

[18] E. Zavala, X. Franch, and J. Marco, "Adaptive monitoring: A systematic mapping," *Inf. Softw. Technol.*, vol. 105, pp. 161–189, Jan. 2019.

[19] J. Gorroñogoitia, D. Valerio, T. Ionescu, and E. Zavala, "D4 . 4: Methods and tools to enact software adaptation and personalization v1," 2016.

[20] A. Perini *et al.*, "D4 . 7: Feedback-gathering and monitoring reconfiguration techniques v1," 2016.

[21] E. Zavala, "Dealing with Uncertainty in Contextual Requirements at Runtime: A Proof of Concept," Universitat Politècnica de Catalunya, 2015.

[22] E. Zavala, X. Franch, J. Marco, A. Knauss, and D. Damian, "SACRE: A tool for dealing with uncertainty in contextual requirements at runtime," in *23rd IEEE International Requirements Engineering Conference (RE)*, 2015, pp. 278–279.

[23] M. Oriol *et al.*, "D4 . 8: Feedback-gathering and monitoring reconfiguration techniques v2," 2016.

[24] J. Gorroñogoitia, E. Zavala, M. Oriol, Q. Motger, and S. Stevanetic, "D4 . 5: Methods and tools to enact software adaptation and personalization v2," 2016.

[25] J. Gorroñogoitia, D. Muñante, F. Kifetew, A. Susi, E. Zavala, and S. Stevanetic, "D3 . 6: Methods and techniques for runtime DM v1," 2016.

[26] E. Zavala, C. Berger, X. Franch, and J. Marco, "Smart self-driving vehicle project: Final report," 2018.

[27] E. Zavala, X. Franch, and J. Marco, "Decision-Making Support for Software Adaptation at Runtime." BSR winter school Big Software on the Run: Where Software meets Data. Tutorials & Poster abstracts. BPM Center Report BPM-16-10, pp. 70–73, 2016.

[28] E. Zavala, "Towards Adaptive Monitoring Services for Self-Adaptive Software Systems," Springer, Cham, 2018, pp. 357–362.

[29] E. Zavala, "Towards Adaptive Monitoring for Self- * Systems: Research Plan," 2017.

[30] G. Liu, M. Trotter, Y. Ren, and T. Wood, "NetAlytics: Cloud-Scale Application Performance Monitoring with SDN and NFV Guyue," in *17th International Middleware Conference (Middleware)*, 2016, pp. 1–14.

[31] T. Kijewski-Correa, M. Haenggi, F. Hall, P. Antsaklis, and F. Hall, "Wireless Sensor Networks for Structural Health Monitoring," *Signal Processing*, vol. 76, no. 12, pp. 1–22, 2006.

[32] H. H. Mshali, T. Lemlouma, and D. Magoni, "Context-Aware Adaptive Framework for e-Health Monitoring," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, 2015, pp. 276–283.

[33] A. Toueir, J. Broisin, and M. Sibilla, "Goal-oriented monitoring adaptation: Methodology and patterns," in *IFIP International Conference on Autonomous Infrastructure, Management and Security (AIMS)*, 2014, vol. 8508 LNCS, pp. 133–146.

[34] C. Alippi, G. Anastasi, C. Galperti, F. Mancini, and M. Rove, "Adaptive Sampling for Energy Conservation in Wireless Sensor Networks for Snow Monitoring Applications," in *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, 2007, pp. 1–6.

[35] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic

mapping studies in software engineering: An update," in *Information and Software Technology*, 2015, vol. 64, pp. 1–18.

[36] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," *Engineering*, vol. 2, p. 1051, 2007.

[37] M. B. Miles, M. a Huberman, and J. Saldana, *Qualitative Data Analysis: A Methods Sourcebook*, 3rd ed. California, USA: SAGE Publications, 2014.

[38] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[39] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining,*. Pearson Publishing, 2005.

[40] Machine Learning Group at the University of Waikato, "Weka 3: Data Mining Software in Java," 2016. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.

[41] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Trans. Auton. Adapt. Syst.*, vol. 4, no. 2, 2009.

[42] P. Bukenya, P. Moyo, H. Beushausen, and C. Oosthuizen, "Health monitoring of concrete dams: A literature review," *J. Civ. Struct. Heal. Monit.*, vol. 4, no. 4, pp. 235–244, 2014.

[43] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic Mapping Studies in Software Engineering," in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, 2008, pp. 68–77.

[44] T. Dybå, T. Dingsøyr, and G. K. Hanssen, "Applying systematic reviews to diverse study types: An experience report," in *Proceedings - 1st International Symposium on Empirical Software Engineering and Measurement, ESEM 2007*, 2007, no. 7465, pp. 225–234.

[45] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009.

[46] F. Magalhães, A. Cunha, and E. Caetano, "Vibration based structural health monitoring of an arch bridge: From automated OMA to damage detection," *Mech. Syst. Signal Process.*, vol. 28, pp. 212–228, 2012.

[47] K. P. Clark, M. Warnier, and F. M. T. Brazier, "Self-adaptive service level agreement monitoring in cloud environments," in *Multiagent and Grid Systems*, 2013, vol. 9, no. 2, pp. 135–155.

[48] D. Jeswani, M. Natu, and R. K. Ghosh, "Adaptive monitoring: A framework to adapt passive monitoring using probing," in *Proceedings of the 2012 8th International Conference on Network and Service Management, CNSM 2012*, 2012, pp. 350–356.

[49] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication

in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, 2014.

[50] E. Zavala, X. Franch, and J. Marco, "Adaptive Monitoring: A systematic Mapping - Studies RQs data, Mendeley Data, v1." Mendeley Data, v1, 2018.

[51] A. Moui *et al.*, "A CIM-based framework to manage monitoring adaptability," in *Proceedings of the 2012 8th International Conference on Network and Service Management, CNSM 2012*, 2012, pp. 261–265.

[52] A. Moui, T. Desprats, E. Lavinal, and M. Sibilla, "Information Models for Managing Monitoring Adaptation Enforcement," *Int. Conf. Adapt. Self-Adaptive Syst. Appl.*, no. c, pp. 44–50, 2012.

[53] O. Franco-Bedoya, D. Ameller, D. Costal, and X. Franch, "Open source software ecosystems: A Systematic mapping," *Inf. Softw. Technol.*, vol. 91, pp. 160–185, 2017.

[54] P. H. Nguyen, S. Ali, and T. Yue, "Model-based security engineering for cyber-physical systems: A systematic mapping study," *Information and Software Technology*, vol. 83. Elsevier, pp. 116–135, 01-Mar-2017.

[55] R. Hoda, N. Salleh, J. Grundy, and H. M. Tee, "Systematic literature reviews in agile software development: A tertiary study," *Inf. Softw. Technol.*, vol. 85, pp. 1339–1351, May 2017.

[56] F. Febrero, C. Calero, and M. Á. Moraga, "A systematic mapping study of software reliability modeling," *Information and Software Technology*, vol. 56, no. 8. Elsevier, pp. 839–849, 01-Aug-2014.

[57] K. R. Felizardo, S. R. S. Souza, and J. C. Maldonado, "The Use of Visual Text Mining to Support the Study Selection Activity in Systematic Literature Reviews: A Replication Study," in *2013 3rd International Workshop on Replication in Empirical Software Engineering Research*, 2013, pp. 91–100.

[58] K. R. Felizardo, N. Salleh, R. M. Martins, E. Mendes, S. G. MacDonell, and J. C. Maldonado, "Using Visual Text Mining to Support the Study Selection Activity in Systematic Literature Reviews," *2011 Int. Symp. Empir. Softw. Eng. Meas.*, pp. 77–86, 2011.

[59] C. Marshall and P. Brereton, "Tools to support systematic literature reviews in software engineering: A mapping study," *Int. Symp. Empir. Softw. Eng. Meas.*, pp. 296–299, 2013.

[60] P. D. Skalski, K. A. Neuendorf, and J. A. Cajigas, "Content Analysis in the Interactive Media Age," in *The Content Analysis Guidebook*, 2017, pp. 201–242.

[61] A. Knauss, D. Damian, X. Franch, A. Rook, H. A. Müller, and A. Thomo, "Acon: A learning-based approach to deal with uncertainty in contextual requirements at runtime," *Inf. Softw. Technol.*, vol. 70, pp. 85–99, 2016.

[62] D. Ameller, X. Burgués, O. Collell, D. Costal, X. Franch, and M. P. Papazoglou,

"Development of service-oriented architectures using model-driven development: A mapping study," *Inf. Softw. Technol.*, vol. 62, no. 1, pp. 42–66, 2015.

[63] F. Ruiz González, "La Investigación en Informática en España : Análisis bibliométrico," *Novatica*, vol. 215, pp. 54–58, 2012.

[64] N. M. Villegas, H. A. Müller, and G. Tamura, "Optimizing run-time SOA governance through context-driven SLAs and dynamic monitoring," in *2011 International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems, MESOCA 2011*, 2011, pp. 1–10.

[65] B. H. C. Cheng *et al.*, "Software Engineering for Self-Adaptive Systems: A Research Roadmap," *Softw. Eng. Self-Adaptive Syst.*, vol. 5525 LNCS, pp. 1–26, 2009.

[66] D. Weyns, "Software Engineering of Self-Adaptive Systems: An Organised Tour and Future Challenges," in *Handbook of Software Engineering*, Springer, 2017.

[67] R. De Lemos *et al.*, "Software engineering for self-adaptive systems: A second research roadmap," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7475 LNCS, pp. 1–32, 2013.

[68] I. Gerostathopoulos, D. Skoda, F. Plasil, T. Bures, and A. Knauss, "Architectural Homeostasis in Self-Adaptive Software-Intensive Cyber-Physical Systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7957, no. January, 2016, pp. 113–128.

[69] V. Klos, T. Gothel, and S. Glesner, "Adaptive Knowledge Bases in Self-Adaptive System Design," *Proc. - 41st Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2015*, pp. 472–478, 2015.

[70] D. Han, J. Xing, Q. Yang, J. Li, and H. Wang, "Handling Uncertainty in Self-Adaptive Software Using Self-Learning Fuzzy Neural Network," *Proc. - Int. Comput. Softw. Appl. Conf.*, vol. 2, no. 1, pp. 540–545, 2016.

[71] I. Gerostathopoulos *et al.*, "Self-adaptation in software-intensive cyber–physical systems: From system goals to architecture configurations," *J. Syst. Softw.*, vol. 122, pp. 378–397, Dec. 2016.

[72] C. Krupitzer, F. M. Roth, S. Vansyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervasive Mob. Comput.*, vol. 17, no. PB, pp. 184–206, 2015.

[73] D. I. K. Sjøberg, T. Dybå, B. C. D. Anda, and J. E. Hannay, "Building theories in software engineering," in *Guide to Advanced Empirical Software Engineering*, 2008, pp. 312–336.

[74] N. M. Villegas, G. Tamura, H. A. Müller, L. Duchien, and R. Casallas, "DYNAMICO: A reference model for governing control objectives and context relevance in self-adaptive software systems," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7475 LNCS, pp. 265–293, 2013.

[75] R. J. Anthony, "Policy-centric integration and dynamic composition of autonomic computing techniques," in *ICAC*, 2007.

[76] R. J. Anthony, "A versatile policy toolkit supporting run-time policy reconfiguration," *Cluster Comput.*, vol. 11, no. 3, pp. 287–298, 2008.

[77] R. Anthony *et al.*, "Autonomic middleware for automotive embedded systems," in *Autonomic Communication*, 2009, pp. 169–210.

[78] E. Lee, Y.-G. G. Kim, Y.-D. D. Seo, K. Seol, and D.-K. K. Baik, "RINGA: Design and verification of finite state machine for self-adaptive software at runtime," *Inf. Softw. Technol.*, vol. 93, no. September 2017, pp. 200–222, 2018.

[79] T. Zhao, "The Generation and Evolution of Adaptation Rules in Requirements Driven Self-adaptive Systems," in *Requirements Engineering Conference (RE), 2016 IEEE 24th International*, 2016.

[80] J. Kramer and J. Magee, *Self-Managed Systems: An Architectural Chal- lenge*. FOSE '07. IEEE Computer So- ciety: In Future of Software Engineering, 2007.

[81] M. U. Iftikhar and D. Weyns, "ActivFORMS: active formal models for self-adaptation," in *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems - SEAMS 2014*, 2014, pp. 125–134.

[82] D. Sykes, D. Corapi, J. Magee, J. Kramer, A. Russo, and K. Inoue, "Learning revised models for planning in adaptive systems," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 63–71.

[83] D. Corapi, D. Sykes, K. Inoue, and A. Russo, "Probabilistic rule learning in nonmonotonic domains," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6814 LNAI, pp. 243–258, 2011.

[84] D. Sykes, W. Heaven, J. Magee, and J. Kramer, "From goals to components: A combined approach to self-management," *SEAMS'08 Proc. 2008 Int. Work. Softw. Eng. Adapt. self-managing Syst.*, pp. 1–8, 2008.

[85] D. Sykes, J. Magee, and J. Kramer, "FlashMob: Distributed Adaptive Self-Assembly," in *Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems - SEAMS '11*, 2011, p. 100.

[86] H. Nakagawa, A. Ohsuga, and S. Honiden, "Towards Dynamic Evolution of Self-Adaptive Systems Based on Dynamic Updating of Control Loops," *2012 IEEE Sixth Int. Conf. Self-Adaptive Self-Organizing Syst.*, pp. 59–68, Sep. 2012.

[87] H. Tajalli, J. Garcia, G. Edwards, and N. Medvidovic, "PLASMA: A plan-based layered architecture for software model-driven adaptation," in *Proceedings of the IEEE/ACM international conference on Automated software engineering - ASE '10*, 2010, p. 467.

[88] N. Medvidovic, D. S. Rosenblum, and R. N. Taylor, "A language and environment for architecture-based software development and evolution," *Proc. 21st Int. Conf. Softw. Eng. - ICSE '99*, pp. 44–53, 1999.

[89] N. Esfahani, A. Elkhodary, and S. Malek, "A learning-based framework for engineering feature-oriented self-adaptive software systems," *IEEE Trans. Softw. Eng.*, vol. 39, no. 11, pp. 1467–1493, Nov. 2013.

[90] A. Elkhodary, N. Esfahani, and S. Malek, "FUSION: A framework for engineering self-tuning self-adaptive software systems," in *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 2010, pp. 7–16.

[91] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli, "Model Evolution by Run-Time Parameter Adaptation," in *Proceedings - International Conference on Software Engineering*, 2009, pp. 111–121.

[92] H. Prothmann, F. Rochner, S. Tomforde, J. Branke, C. Müller-Schloer, and H. Schmeck, "Organic control of traffic lights," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5060 LNCS, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 219–233.

[93] J. Branke *et al.*, "Organic Computing - Addressing complexity by controlled self-organization," *Proc. - ISoLA 2006 2nd Int. Symp. Leveraging Appl. Form. Methods, Verif. Valid.*, pp. 185–191, 2007.

[94] F. Rochner, H. Prothmann, J. Branke, C. Müller-Schloer, and H. Schmeck, "An organic architecture for traffic light controllers," *Proc. Inform.*, vol. 1, pp. 120–127, 2006.

[95] J. Branke, P. Goldate, and H. Prothmann, "Actuated traffic signal optimization using evolutionary algorithms," in *Proceedings of the 6th European Congress and Exhibition on Intelligent Transport Systems and Services (ITS 2007)*, 2007.

[96] S. Tomforde *et al.*, "Decentralised progressive signal systems for organic traffic control," in *Proceedings - 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2008*, 2008, pp. 413–422.

[97] F. M. Roth, C. Krupitzer, and C. Becker, "Runtime evolution of the adaptation logic in self-adaptive systems," in *Proceedings - IEEE International Conference on Autonomic Computing, ICAC 2015*, 2015, pp. 141–142.

[98] A. M. Sharifloo, A. Metzger, C. Quinton, L. Baresi, and K. Pohl, "Learning and evolution in dynamic software product lines," in *Proceedings of the 11th International Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '16*, 2016, pp. 158–164.

[99] P. Jamshidi, A. M. Sharifloo, C. Pahl, A. Metzger, and G. Estrada, "Self-Learning Cloud Controllers: Fuzzy Q-Learning for Knowledge Evolution," *Proc. - 2015 Int. Conf. Cloud Auton. Comput. ICCAC 2015*, pp. 208–211, 2015.

[100] L. Baresi and C. Quinton, "Dynamically Evolving the Structural Variability of Dynamic Software Product Lines," *Proc. - 10th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst. SEAMS 2015*, pp. 57–63, 2015.

[101] C. Quinton, R. Rabiser, M. Vierhauser, P. Grünbacher, and L. Baresi, "Evolution in

dynamic software product lines: challenges and perspectives," *Proc. 19th Int. Conf. Softw. Prod. Line - SPLC '15*, pp. 126–130, 2015.

[102] L. Pasquale, L. Baresi, and B. Nuseibeh, "Towards adaptive systems through requirements@runtime?," in *CEUR Workshop Proceedings*, 2011, vol. 794, pp. 13–24.

[103] L. Baresi, L. Pasquale, and P. Spoletini, "Fuzzy goals for requirements-driven adaptation," in *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference (RE'10)*, 2010, pp. 125–134.

[104] L. Baresi and L. Pasquale, "An eclipse plug-in to model system requirements and adaptation capabilities," in *Proc. of the 6th IT-Eclipse Workshop*, 2011.

[105] C. Dorn and S. Dustdar, "Interaction-driven self-adaptation of service ensembles," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6051 LNCS, pp. 393–408, 2010.

[106] C. Dorn, D. Schall, and S. Dustdar, "Context-aware adaptive service mashups," *2009 IEEE Asia-Pacific Serv. Comput. Conf. APSCC 2009*, no. c, pp. 301–306, 2009.

[107] Z. A. Mann and A. Metzger, "Auto-Adjusting Self-Adaptive Software Systems," in *2018 IEEE International Conference on Autonomic Computing (ICAC)*, 2018, pp. 181–186.

[108] "ActivFORMS: Active Formal Models for Selfadaptation."

[109] C. Krupitzer, F. M. Roth, S. Vansyckel, and C. Becker, "Towards reusability in autonomic computing," in *Proceedings - IEEE International Conference on Autonomic Computing, ICAC 2015*, 2015, pp. 115–120.

[110] C. Krupitzer, S. Vansyckel, and C. Becker, "FESAS: Towards a Framework for Engineering Self-Adaptive Systems," in *2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*, 2013, pp. 263–264.

[111] V. Krishnasree, N. Balaji, and P. Sudhakar Rao, "A real time improved driver fatigue monitoring system," *WSEAS Trans. Signal Process.*, vol. 10, no. 1, pp. 146–155, 2014.

[112] A. Sahayadhas, K. Sundaraj, and M. Murugappan, "Detecting driver drowsiness based on sensors: A review," *Sensors (Switzerland)*, vol. 12, no. 12, pp. 16937–16953, 2012.

[113] J. Lee, J. Choi, K. Yi, M. Shin, and B. Ko, "Lane-keeping assistance control algorithm using differential braking to prevent unintended lane departures," *Control Eng. Pract.*, vol. 23, no. 1, pp. 1–13, 2014.

[114] J. Lisseman, D. Andrews, and J. Bosch, "Steering wheel with hand pressure sensing," 2015.

[115] L. M. Bergasa, J. Nuevo, M. A. Sotelo, R. Barea, and M. E. Lopez, "Real-Time System for Monitoring Driver Vigilance," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 63–77, 2006.

[116] W. C. Liang, J. Yuan, D. C. Sun, and M. H. Lin, "Changes in physiological parameters

induced by indoor simulated driving: Effect of lower body exercise at mid-term break," *Sensors*, vol. 9, no. 9, pp. 6913–6933, 2009.

[117] T. Wartzek, B. Eilebrecht, J. Lem, H. J. Lindner, S. Leonhardt, and M. Walter, "ECG on the road: Robust and unobtrusive estimation of heart rate," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 11, pp. 3112–3120, 2011.

[118] BMW, "Steering and lane control assitant incl. traffic jam assitant.," 2016. [Online]. Available: http://www.bmw.com/com/en/insights/technology/connecteddrive/2013/driver_assistance/intelligent_driving.html.

[119] U. S. A. Toyota Motor Sales, "Toyota Safety Sense P (TSS P) - Pre Collision System with Pedestrian Detection (PCS)," 2016. [Online]. Available: http://www.toyota.com/safety-sense.

[120] J. Jiménez-Pinto and M. Torres-Torriti, "Optical flow and driver's kinematics analysis for state of alert sensing.," *Sensors (Basel).*, vol. 13, no. 4, pp. 4225–4257, 2013.

[121] M. L. Berenson and D. M. Levine, *Basic Bussiness Statistics: Concepts and Applications*, 6th ed. Prentice-Hall International, Inc., 1996.

[122] T. Baumhöfer, M. Brühl, S. Rothgang, and D. U. Sauer, "Production caused variation in capacity aging trend and correlation to initial cell performance," *J. Power Sources*, vol. 247, pp. 332–338, 2014.

[123] A. Rook, A. Knauss, D. Damian, and A. Thomo, "A Case Study of Applying Data Mining to Sensor Data for Contextual Requirements Analysis," *2014 IEEE 1st Int. Work. Artif. Intell. Requir. Eng.*, pp. 43–50, 2014.

[124] A. Rook, "On the Feasibility of Integrating Data Mining Algorithms into Self Adaptive Systems for Context Awareness and Requirements Evolution (Master thesis)," University of Victoria, 2014.

[125] C. Berger, "From a Competition for Self-Driving Miniature Cars to a Standardized Experimental Platform: Concept, Models, Architecture, and Evaluation," *J. Softw. Eng. Robot.*, vol. 5, no. 1, pp. 63–79, Jun. 2014.

[126] C. Berger, "An open continuous deployment infrastructure for a self-driving vehicle ecosystem," *IFIP Adv. Inf. Commun. Technol.*, vol. 472, pp. 177–183, 2016.

[127] ETSI, "Intelligent Transport Systems (ITS) - Vehicular Communications - Basic Set of Applications - Part 2 : Specification of Cooperative Awareness Basic Service," *History*, vol. 1, pp. 1–22, 2011.

[128] ETSI, "ETSI EN 302 637-3 Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service," *Etsi*, vol. 1, pp. 1–73, 2010.

[129] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, 1991.

A

APPENDIX

How to adapt: Study on adaptive monitoring

A1 SMS references

[R1] Aderohunmu, F. A., Paci, G., Benini, L., Deng, J. D., & Brunelli, D. (2013). SWIFTNET: A data acquisition protocol for fast-reactive monitoring applications. In *IEEE International Symposium on Industrial Embedded Systems (SIES)* (pp. 93–96). IEEE. https://doi.org/10.1109/SIES.2013.6601478

[R2] Agarwala, S., Chen, Y., Milojicic, D., & Schwan, K. (2006). QMON: QoS-and utility-aware monitoring in enterprise systems. In *IEEE International Conference on Autonomic Computing (ICAC)* (Vol. 2006, p. 124-133).

[R3] Alippi, C., Anastasi, G., Galperti, C., Mancini, F., & Rove, M. (2007). Adaptive Sampling for Energy Conservation in Wireless Sensor Networks for Snow Monitoring Applications. In *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)* (pp. 1–6). https://doi.org/10.1109/MOBHOC.2007.4428700

[R4] Allman, M., & Paxson, V. (2008). A reactive measurement framework. In *International Conference on Passive and Active Network Measurement (PAM)* (Vol. 4979 LNCS, pp. 92–101). https://doi.org/10.1007/978-3-540-79232-1_10

[R5] Arumuga Nainar, P., & Liblit, B. (2010). Adaptive bug isolation. In *32nd ACM/IEEE International Conference on Software Engineering (ICSE)* (Vol. 1, p. 255). New York, NY, USA. https://doi.org/10.1145/1806799.1806839

[R6] Baresi, L., & Guinea, S. (2005). Towards dynamic monitoring of WS-BPEL processes. In *International Conference on Service-Oriented Computing (ICSOC)* (Vol. 3826 LNCS, pp. 269–282). https://doi.org/10.1007/11596141_21

[R7] Barford, P., Duffield, N., Ron, A., & Sommers, J. (2009). Network Performance Anomaly Detection and Localization. *28th IEEE INFOCOM Conference on Computer Communications*, 1377–1385. https://doi.org/10.1109/INFCOM.2009.5062053

[R8] Batalin, M. A., Srivastava, M., Estrin, D., Rahimi, M., Yu, Y., Liu, D., … Pottie, G. J. (2004). Call and response. In *2nd international conference on Embedded networked sensor systems (SenSys)* (p. 25). https://doi.org/10.1145/1031495.1031499

[R9] Bertolino, A., Calabrò, A., Lonetti, F., Di Marco, A., & Sabetta, A. (2011). Towards a model-driven infrastructure for runtime monitoring. In *International Workshop on Software Engineering for Resilient Systems (SERENE)* (Vol. 6968 LNCS, pp. 130–144). https://doi.org/10.1007/978-3-642-24124-6_13

[R10] Bhatia, S., Kumar, A., Fiuczynski, M. E., & Peterson, L. (2008). Lightweight, High-Resolution Monitoring for Troubleshooting Production Systems. In *8th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (pp. 103--116). https://doi.org/10.1.1.145.5057

[R11] Brodie, M., Rish, I., Ma, S., Odintsova, N., & Beygelzimer, A. (2003). Active Probing Strategies for Problem Diagnosis in Distributed Systems. In *18th International joint conference on Artificial intelligence (IJCAI)* (pp. 1337–1338).

[R12] Cantieni, G. R., Iannaccone, G., Barakat, C., Diot, C., & Thiran, P. (2006). Reformulating the monitor placement problem: Optimal network-wide sampling. In *IEEE Conference on Information Sciences and Systems (CISS)* (pp. 1725–1731). https://doi.org/10.1109/CISS.2006.286433

[R13] Chen, Q., Wang, L., & Yang, Z. (2012). SAM: Self-adaptive dynamic analysis for multithreaded programs. In *Haifa Verification Conference (HVC)* (Vol. 7261 LNCS, pp. 115–129). https://doi.org/10.1007/978-3-642-34188-5_12

[R14] Chu, M., Haussecker, H., & Feng Zhao. (2002). Scalable Information-Driven Sensor Querying and Routing for Ad Hoc Heterogeneous Sensor Networks. In *International Journal of High Performance Computing Applications* (Vol. 16, pp. 293–313). https://doi.org/10.1177/109434200201600309 01

[R15] Clark, K. P., Warnier, M., & Brazier, F. M. T. (2013). Self-adaptive service level agreement monitoring in cloud environments. In *Multiagent and Grid Systems* (Vol. 9, pp. 135–155). https://doi.org/10.3233/MGS-130203

[R16] Clark, K., Warnier, M., & Brazier, F. M. T. (2011). Self-adaptive service monitoring. In *Adaptive and Intelligent Systems* (Vol. 6943 LNAI, pp. 119–130). https://doi.org/10.1007/978-3-642-23857-4_15

[R17] Comuzzi, M., & Spanoudakis, G. (2010). Dynamic set-up of monitoring infrastructures for service based systems. In *ACM Symposium on Applied Computing (SAC)* (p. 2414). https://doi.org/10.1145/1774088.1774591

[R18] Contreras, R., & Zisman, A. (2011). Identifying, modifying, creating, and removing monitor rules for service oriented computing. In *3rd international workshop on Principles of engineering service-oriented systems (PESOS)* (pp. 43–49). New York, NY, USA: ACM. https://doi.org/10.1145/1985394.1985401

[R19] Contreras, R., & Zisman, A. (2010). A pattern-based approach for monitor adaptation. In *IEEE International Conference on Software Science, Technology, and Engineering (SwSTE)* (pp. 30–37). IEEE. https://doi.org/10.1109/SwSTE.2010.12

[R20] Cotroneo, D., Di Leo, D., & Natella, R. (2010). Adaptive monitoring in microkernel OSs. In *International Conference on Dependable Systems and Networks Workshops (DSN-W)* (pp. 66–72). IEEE. https://doi.org/10.1109/DSNW.2010.5542619

[R21] Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., & Hong, W. (2004). Model-Driven Data Acquisition in Sensor Networks. In *30th International conference on Very large data bases (VLDB)* (pp. 588–599). Elsevier. https://doi.org/10.1016/B978-012088469-8/50053-X

[R22] Deshpande, A., Guestrin, C., Madden, S. R., Hellerstein, J. M., & Hong, W. (2005). Model-based approximate querying in sensor networks. *International Journal on Very Large Data Bases (VLDB Journal)*, *14*(4), 417–443. https://doi.org/10.1007/s00778-005-0159-3

[R23] Dilman, M., & Raz, D. (2001). Efficient reactive monitoring. In *20th IEEE INFOCOM Conference on Computer Communications* (Vol. 20, pp. 668–676). https://doi.org/10.1109/JSAC.2002.1003034

[R24] Dmitriev, M. (2004). Profiling Java applications using code hot swapping and dynamic call graph revelation. In *4th International workshop on Software and performance (WOSP)* (Vol. 29, p. 139). New York, New York, USA: ACM Press. https://doi.org/10.1145/974044.974067

[R25] Doelitzscher, F., Reich, C., Knahl, M., Passfall, A., & Clarke, N. (2012). An agent based business aware incident detection system for cloud environments. *Journal of Cloud Computing: Advances, Systems and Applications*, *1*(1), 9. https://doi.org/10.1186/2192-113X-1-9

[R26] Dwyer, M. B., Kinneer, A., & Elbaum, S. (2007). Adaptive Online Program Analysis. In *29th ACM/IEEE International Conference on Software Engineering (ICSE)* (pp. 220–229). IEEE. https://doi.org/10.1109/ICSE.2007.12

[R27] Ehlers, J., & Hasselbring, W. (2011). A self-adaptive monitoring framework for component-based software systems. In *European Conference on Software Architecture (ECSA)* (Vol. 6903 LNCS, pp. 278–286). https://doi.org/10.1007/978-3-642-23798-0_30

[R28] Ehlers, J., van Hoorn, A., Waller, J., & Hasselbring, W. (2011). Self-adaptive software system monitoring for performance anomaly localization. In *8th IEEE international conference on Autonomic computing (ICAC)* (p. 197). New York, NY, USA: ACM. https://doi.org/10.1145/1998582.1998628

[R29] Estan, C., Keys, K., Moore, D., & Varghese, G. (2004). Building a better NetFlow. In *Conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)* (Vol. 135, p. 245). New York, New York, USA: ACM Press. https://doi.org/10.1145/1015467.1015495

[R30] Fan Ye, Zhong, G., Cheng, J., Songwu Lu, & Lixia Zhang. (2003). PEAS: a robust energy conserving protocol for long-lived sensor networks. In *23rd International Conference on Distributed Computing Systems (ICDCS)* (pp. 28–37). IEEE. https://doi.org/10.1109/ICDCS.2003.1203449

[R31] Fan, L., & Xiong, L. (2012). Real-time aggregate monitoring with differential privacy. In *21st ACM international conference on Information and knowledge management (CIKM)* (Vol. 26, p. 2169). New York, New York, USA: ACM Press. https://doi.org/10.1145/2396761.2398595

[R32] Fei, L., & Midkiff, S. (2006). Artemis: Practical runtime monitoring of applications for execution anomalies. In *ACM SIGPLAN Conference on Programming Language*

Design and Implementation (PLDI) (Vol. 41, pp. 84–95). https://doi.org/10.1145/1133981.1133992

[R33] Feng Zhao, Jaewon Shin, & Reich, J. (2002). Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine*, *19*(2), 61–72. https://doi.org/10.1109/79.985685

[R34] Findrik, M., Kristensen, T. le F., Hinterhofer, T., Olsen, R. L., & Schwefel, H. P. (2015). Information-quality based LV-grid-monitoring framework and its application to power-quality control. In *International Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW)* (Vol. 9143 LNCS, pp. 317–329). https://doi.org/10.1007/978-3-319-19662-6_22

[R35] Gedik, B. B. B., Liu, L., & Yu, P. S. (2007). ASAP: An Adaptive Sampling Approach to Data Collection in Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems*, *18*(12), 1766–1783. https://doi.org/10.1109/TPDS.2007.1110

[R36] Gonzalez-Herrera, I., Bourcier, J., Daubert, E., Rudametkin, W., Barais, O., Fouquet, F., … Baudry, B. (2016). ScapeGoat: Spotting abnormal resource usage in component-based reconfigurable software systems. *Journal of Systems and Software*, *122*, 398–415. https://doi.org/10.1016/j.jss.2016.02.027

[R37] Gonzalez-Herrera, I., Bourcier, J., Daubert, E., Rudametkin, W., Barais, O., Fouquet, F., & Jezequel, J. M. (2014). Scapegoat: An adaptive monitoring framework for component-based systems. In *IEEE/IFIP Conference on Software Architecture (WICSA)* (pp. 67–76). IEEE. https://doi.org/10.1109/WICSA.2014.49

[R38] Groenendijk, J., Huang, Y., & Fallon, L. (2011). Adaptive Terminal Reporting for Scalable Service Quality Monitoring in Large Networks. In *7th International Conference on Network and Service Management (CNSM)* (pp. 427–431).

[R39] Halal, F., Pedrocca, P., Hirose, T., Cretu, A. M., & Zaremba, M. B. (2014). Remote-sensing based adaptive path planning for an aquatic platform to monitor water quality. In *IEEE International Symposium on RObotic and SEnsors Environments (ROSE)* (pp. 43–48). IEEE. https://doi.org/10.1109/ROSE.2014.6952981

[R40] Haran, M., Karr, A., Last, M., Orso, A., Porter, A. A., Sanil, A., & Fouche, S. (2007). Techniques for classifying executions of deployed software to support software engineering tasks. *IEEE Transactions on Software Engineering*, *33*(5), 287–304. https://doi.org/10.1109/TSE.2007.1004

[R41] Hernandez, E. a, Chidester, M. C., & George, A. D. (2001). Adaptive Sampling for Network Management. *Journal of Network and Systems Management*, *9*(4), 409–434. https://doi.org/10.1023/A:1012980307500

[R42] Horling, B., Vincent, R., Mailler, R., Shen, J., Becker, R., Rawlins, K., & Lesser, V. (2001). Distributed sensor network for real time tracking. In *5th international conference on Autonomous agents* (pp. 417–424).

[R43] Iacono, M., Romano, E., & Marrone, S. (2010). Adaptive monitoring of marine disasters with intelligent mobile sensor networks. In *IEEE Workshop on Environmental Energy and Structural Monitoring Systems (EESMS)* (pp. 38–45). IEEE. https://doi.org/10.1109/EESMS.2010.5634179

[R44] Jain, A., & Chang, E. Y. (2004). Adaptive sampling for sensor networks. In *1st International workshop on Data Management for Sensor Networks (DMSN): in conjunction with the International Conference on Very Large Data Bases (VLDB)* (pp. 10–16). https://doi.org/10.1145/1052199.1052202

[R45] Jeswani, D., Natu, M., & Ghosh, R. K. (2012). Adaptive Monitoring: A Framework to Adapt Passive Monitoring using Probing. *8th International Conference on Network and Service Management (Cnsm) and Workshop on Systems Virtualiztion Management (Svm)*, 350–356. https://doi.org/10.1007/s10922-014-9330-8

[R46] Jeswani, D., Natu, M., & Ghosh, R. K. (2015). Adaptive Monitoring: Application of Probing to Adapt Passive Monitoring. *Journal of Network and Systems Management*, *23*(4), 950–977. https://doi.org/10.1007/s10922-014-9330-8

[R47] Ji, X., Zha, H., Metzner, J. J., & Kesidis, G. (2004). Dynamic Cluster Structure for Object Detection and Tracking in Wireless Ad-Hoc Sensor Networks. In *IEEE International Conference on Communications* (Vol. 0, pp. 3807–3811).

[R48] Jiao, J., Naqvi, S., Raz, D., & Sugla, B. (2000). Toward efficient monitoring. *IEEE Journal on Selected Areas in Communications*, *18*(5), 723–732. https://doi.org/10.1109/49.842988

[R49] Katsaros, G., Kousiouris, G., Gogouvitis, S. V., Kyriazis, D., Menychtas, A., & Varvarigou, T. (2012). A Self-adaptive hierarchical monitoring mechanism for Clouds. *Journal of Systems and Software*, *85*(5), 1029–1041. https://doi.org/10.1016/j.jss.2011.11.1043

[R50] Kho, J., Rogers, A., & Jennings, N. R. (2007). Decentralised Adaptive Sampling of Wireless Sensor Networks. In *1st International Workshop on Agent Technology for Sensor Networks at the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (Vol. 5, pp. 55–62).

[R51] Kiciman, E., & Livshits, B. (2007). AjaxScope: a platform for remotely monitoring the client-side behavior of Web 2.0 applications. In *ACM SIGOPS symposium on Operating systems principles (SOSP)* (Vol. 41, pp. 17–30). https://doi.org/http://doi.acm.org/10.1145/1294261.1294264

[R52] Kim, H., Yoon, H., Cho, Y., Park, S., & Sugumaran, V. (2011). Multi-layered adaptive monitoring in service robots. In *5th International Conference on Secure Software Integration and Reliability Improvement - Companion (SSIRI-C)* (pp. 76–83). IEEE. https://doi.org/10.1109/SSIRI-C.2011.22

[R53] Kim, S., & Pakzad, S. (2006). Wireless Sensor Networks for Structural Health Monitoring: A Multi-Scale Approach. In *17th Analysis and Computation Specialty Conference at Structures Congress (ASCE)*. https://doi.org/10.1145/1182807.1182889

[R54] Kyriazis, D., Kostantos, K., Kapsalis, A., Gogouvitis, S., & Varvarigou, T. (2013). QoS-oriented service management in large scale federated clouds. In *IEEE International Symposium on Computers and Communications (ISCC)* (pp. 22–27). IEEE. https://doi.org/10.1109/ISCC.2013.6754917

[R55] Lassoued, I., & Barakat, C. (2011). A Multi-task Adaptive Monitoring System Combining Different Sampling Primitives. In *International Teletraffic Congress (ITC)* (pp. 79–86). International Teletraffic Congress. Retrieved from http://dl.acm.org/citation.cfm?id=2043468.2043482

[R56] Lee, C. G., & Lee, K. S. (2012). A development framework toward reconfigurable run-time monitors. In *IT Convergence and Services* (Vol. 107 LNEE, pp. 519–525). https://doi.org/10.1007/978-94-007-2598-0_55

[R57] Liu, G., Trotter, M., Ren, Y., & Wood, T. (2016). NetAlytics: Cloud-Scale Application Performance Monitoring with SDN and NFV Guyue. In *17th International Middleware Conference (Middleware)* (pp. 1–14). New York, New York, USA: ACM Press.

https://doi.org/10.1145/2988336.2988344

[R58] Liu, J., Guibas, L., & Zhao, F. (2002). A Dual-Space Approach to Tracking and Sensor Management in Wireless Sensor Networks. In *ACM Workshop on Wireless Sensor Networks and Applications* (pp. 131–139). https://doi.org/10.1145/570753.570757

[R59] Madden, S., Franklin, M. J., Hellerstein, J. M., & Berkeley, U. C. (2003). The Design of an Acquisitional Query Processor For Sensor Networks. *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 491–502. https://doi.org/10.1145/872757.872817

[R60] Mainland, G., Parkes, D. C., & Welsh, M. (2005). Decentralized, adaptive resource allocation for sensor networks. In *2nd conference on Symposium on Networked Systems Design & Implementation (NSDI)* (pp. 315–328).

[R61] Massie, M. L., Chun, B. N., & Culler, D. E. (2004). The ganglia distributed monitoring system: Design, implementation, and experience. *Parallel Computing*, *30*(7), 817–840. https://doi.org/10.1016/j.parco.2004.04.001

[R62] Maurel, Y., Bottaro, A., Kopetz, R., & Attouchi, K. (2012). Adaptive monitoring of end-user OSGi-based home boxes. In *15th ACM SIGSOFT symposium on Component Based Software Engineering (CBSE)* (p. 157). New York, NY, USA: ACM. https://doi.org/10.1145/2304736.2304763

[R63] Meng, S., & Liu, L. (2013). Enhanced monitoring-as-a-service for effective cloud management. *IEEE Transactions on Computers*, *62*(9), 1705–1720. https://doi.org/10.1109/TC.2012.165

[R64] Merghem, L., Gaiti, D., & Pujolle, G. (2003). On using multi-agent systems in end to end adaptive monitoring. *IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS)*, *2839*, 422–435.

[R65] Mos, A. (2004). COMPAS: adaptive performance monitoring of component-based systems. In *Workshop on Remote Analysis and Measurement of Software Systems (RAMSS) at 26th ACM/IEEE International Conference on Software Engineering (ICSE)* (Vol. 2004, pp. 35–39). https://doi.org/10.1049/ic:20040348

[R66] Moui, A., & Desprats, T. (2011). Towards self-adaptive monitoring framework for integrated management. In *IFIP International Conference on Autonomous Infrastructure, Management and Security (AIMS)* (Vol. 6734 LNCS, pp. 160–163). https://doi.org/10.1007/978-3-642-21484-4_18

[R67] Moui, A., Desprats, T., Lavinal, E., & Sibilla, M. (2012). A CIM-based framework to manage monitoring adaptability. In *8th international conference on network and service management (cnsm) and workshop on systems virtualiztion management (svm)* (pp. 261–265). Laxenburg, Austria, Austria: IEEE.

[R68] Moui, A., Desprats, T., Lavinal, E., & Sibilla, M. (2012). Information Models for Managing Monitoring Adaptation Enforcement. *International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE)*, (c), 44–50.

[R69] Moui, A., Desprats, T., Lavinal, E., & Sibilla, M. (2010). Managing polling adaptability in a CIM/WBEM infrastructure. In *4th International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM)* (pp. 1–6). https://doi.org/10.1109/SVM.2010.5674749

[R70] Mshali, H. H., Lemlouma, T., & Magoni, D. (2016). Context-Aware Adaptive Framework for e-Health Monitoring. In *IEEE International Conference on Data Science and Data Intensive Systems (DSDIS)* (pp. 276–283). USA: IEEE. https://doi.org/10.1109/DSDIS.2015.13

[R71] Munawar, M. A., Reidemeister, T., Jiang, M., George, A., & Ward, P. A. S. (2008). Adaptive Monitoring with Dynamic Differential Tracing-Based Diagnosis. In *International Workshop on Distributed Systems: Operations and Management (DSOM)* (Vol. 5273 LNCS, pp. 162–175). https://doi.org/10.1007/978-3-540-87353-2_13

[R72] Munawar, M. A., & Ward, P. A. S. (2006). Adaptive monitoring in enterprise software systems. *Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, 1–5.

[R73] Munawar, M. A., & Ward, P. A. S. (2007). Leveraging Many Simple Statistical Models to Adaptively Monitor Software Systems. In I. Stojmenovic, R. K. Thulasiram, L. T. Yang, W. Jia, M. Guo, & R. F. de Mello (Eds.), *International Symposium on Parallel and Distributed Processing and Applications (ISPA)* (Vol. 4742 LNCS, pp. 457–470). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-74742-0_42

[R74] Natu, M., & Sethi, A. S. (2006). Active Probing Approach for Fault Localization in Computer Networks. In *4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services* (pp. 25–33). IEEE. https://doi.org/10.1109/E2EMON.2006.1651276

[R75] Natu, M., & Sethi, A. S. (2007). Probabilistic Fault Diagnosis Using Adaptive Probing. In *International Workshop on Distributed Systems: Operations and Management (DSOM)* (Vol. 4785 LNCS, pp. 38–49). https://doi.org/10.1007/978-3-540-75694-1_4

[R76] Natu, M., & Sethi, A. S. (2008). Application of adaptive probing for fault diagnosis in computer networks. In *IEEE/IFIP Network Operations and Management Symposium: Pervasive Management for Ubiquitous Networks and Services (NOMS)* (pp. 1055–1060). https://doi.org/10.1109/NOMS.2008.4575278

[R77] Natu, M., & Sethi, A. S. (2007). Efficient probing techniques for fault diagnosis. In *2nd International Conference on Internet Monitoring and Protection (ICIMP)* (pp. 0–5). https://doi.org/10.1109/ICIMP.2007.14

[R78] Newman, H. B., Legrand, I. C., Galvez, P., Voicu, R., & Cirstoiu, C. (2003). MonALISA : A Distributed Monitoring Service Architecture. In *13th International Conference on Computing in High-Enery and Nuclear Physics (CHEP)*. https://doi.org/10.1109/IEMBS.2005.1616960

[R79] Nguyen, T. A. B., Siebenhaar, M., Hans, R., & Steinmetz, R. (2014). Role-based templates for cloud monitoring. In *7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)* (pp. 242–250). IEEE. https://doi.org/10.1109/UCC.2014.33

[R80] Nobre, J.C., Granville, L.Z., Clemm, A., Prieto, A. G. G. (2012). Decentralized detection of SLA violations using P2P technology. In *International Conference on Network and Service Management (CNSM)* (pp. 100–107).

[R81] Okanovic, D., van Hoorn, A., Konjovic, Z., Vidakovic, M. (2011). Towards Adaptive Monitoring of Java EE Applications. In *5th International Conference on Information Technology (ICIT)*.

[R82] Okanovic, D., van Hoorn, A., Konjovic, Z., Vidakovic, M., Okanović, D., van Hoorn, A., … Vidakovic, M. (2013). SLA-Driven adaptive monitoring of distributed applications for performance problem localization. *Computer Science and Information Systems*, *10*(1), 25–50. https://doi.org/10.2298/CSIS1109260370

[R83] Orso, A., Liang, D., Harrold, M. J., & Lipton, R. (2002). Gamma System: Continuous Evolution of Software after Deployment. In *ACM SIGSOFT international symposium on Software testing and analysis (ISSTA)* (Vol. 27, p. 65). New York, New York, USA: ACM Press. https://doi.org/10.1145/566172.566182

[R84] Padhy, P., Dash, R. K., Martinez, K., & Jennings, N. R. (2006). A utility-based sensing and communication model for a glacial sensor network. In *5th International joint conference on Autonomous agents and multiagent systems (AAMAS)* (p. 1353). https://doi.org/10.1145/1160633.1160885

[R85] Psiuk, M., & Zielinski, K. (2015). Goal-driven adaptive monitoring of SOA systems. *Journal of Systems and Software*, *110*, 101–121. https://doi.org/10.1016/j.jss.2015.08.015

[R86] Rabiser, R., Vierhauser, M., & Grünbacher, P. (2015). Variability Management for a Runtime Monitoring Infrastructure. In *9th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS)* (pp. 35–42). New York, New York, USA: ACM Press. https://doi.org/10.1145/2701319.2701330

[R87] Rahimi, M., Hansen, M., Kaiser, W. J., Sukhatme, G. S., & Estrin, D. (2005). Adaptive sampling for environmental field estimation using robotic sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3692–3698). IEEE. https://doi.org/10.1109/IROS.2005.1545070

[R88] Rahimi, M., Pon, R., Kaiser, W. J., Sukhatme, G. S., Estrin, D., & Srivastava, M. (2004). Adaptive sampling for environmental robotics. In *IEEE International Conference on Robotics and Automation (ICRA)* (Vol. 4, p. 3537–3544 Vol.4). https://doi.org/10.1109/ROBOT.2004.1308801

[R89] Ramirez, A. J., Cheng, B. H. C., & McKinley, P. K. (2010). Adaptive monitoring of software requirements. In *1st International Workshop on Requirements@Run.Time (RE@RunTime)* (pp. 41–50). IEEE. https://doi.org/10.1109/RE@RUNTIME.2010.5628549

[R90] Rish, I., Brodie, M., Odintsova, N., Sheng Ma, & Grabarnik, G. (2004). Real-time problem determination in distributed systems using active probing. In *IEEE/IFIP Network Operations and Management Symposium* (Vol. 1, pp. 133–146). IEEE. https://doi.org/10.1109/NOMS.2004.1317650

[R91] Rish, I., Brodie, M., Ma, S., Odintsova, N., Beygelzimer, A., Grabarnik, G., & Hernandez, K. (2005). Adaptive diagnosis in distributed systems. *IEEE Transactions on Neural Networks*, *16*(5), 1088–1109. https://doi.org/10.1109/TNN.2005.853423

[R92] Shamsi, J., & Brockmeyer, M. (2012). Predictable service overlay networks: Predictability through adaptive monitoring and efficient overlay construction and management. *Journal of Parallel and Distributed Computing*, *72*(1), 70–82. https://doi.org/10.1016/j.jpdc.2011.09.005

[R93] Shao, J., Wei, H., Wang, Q., & Mei, H. (2010). A Runtime Model Based Monitoring Approach for Cloud. In IEEE (Ed.), *3rd IEEE International Conference on Cloud Computing* (pp. 313–320). IEEE. https://doi.org/10.1109/CLOUD.2010.31

[R94] Shen, D., Tse, K. H., & Chan, C. K. (2012). Adaptive fault monitoring in all-optical networks utilizing real-time data traffic. *Journal of Network and Systems Management*, *20*(1), 76–96. https://doi.org/10.1007/s10922-011-9206-0

[R95] Talwar, V., Shankar, C. S., Rafael, R., Milojicic, D., Iyer, S., Farkas, K., & Chen, Y. (2006). Adaptive Monitoring: Automated Change Management for Monitoring

Systems. In *Workshop of the HP OpenView University Association (HP-OVUA)*.

[R96] Tang, Y. T. Y., Al-Shaer, E. S., & Boutaba, R. (2005). Active integrated fault localization in communication networks. In *9th IFIP/IEEE International Symposium on Integrated Network Management (IM)* (pp. 543–556). https://doi.org/10.1109/INM.2005.1440826

[R97] Thongtra, P., & Aagesen, F. A. (2010). An adaptable capability monitoring system. In *6th International Conference on Networking and Services (ICNS)* (pp. 73–80). https://doi.org/10.1109/ICNS.2010.19

[R98] Tierney, B., Crowley, B., Gunter, D., Lee, J., & Thompson, M. (2000). A Monitoring Sensor Management System for Grid Environments. *9th International Symposium on High-Performance Distributed Computing*, *4*(1), 19–28. https://doi.org/10.1023/A:1011408108941

[R99] Toueir, A., Broisin, J., & Sibilla, M. (2013). A goal-oriented approach for adaptive SLA monitoring: A cloud provider case study. In *2nd IEEE Latin American Conference on Cloud Computing and Communications (LatinCloud)* (pp. 53–58). IEEE. https://doi.org/10.1109/LatinCloud.2013.6842223

[R100] Toueir, A., Broisin, J., & Sibilla, M. (2014). Goal-oriented monitoring adaptation: Methodology and patterns. In *IFIP International Conference on Autonomous Infrastructure, Management and Security (AIMS)* (Vol. 8508 LNCS, pp. 133–146). https://doi.org/10.1007/978-3-662-43862-6_17

[R101] Trihinas, D., Pallis, G., & Dikaiakos, M. D. (2015). AdaM: An adaptive monitoring framework for sampling and filtering on IoT devices. In *IEEE International Conference on Big Data (Big Data)* (pp. 717–726). IEEE. https://doi.org/10.1109/BigData.2015.7363816

[R102] Tseng, Y. C., Wang, Y. C., Cheng, K. Y., & Hsieh, Y. Y. (2007). iMouse: An integrated mobile surveillance and wireless sensor system. *Computer*, *40*(6), 60–67. https://doi.org/10.1109/MC.2007.211

[R103] Villegas, N. M., & Müller, H. A. (2010). Context-Driven Adaptive Monitoring for Supporting SOA Governance. In *International Workshop on a Research Agenda for Maintenance and Evolution of Service-Oriented Systems (MESOA)* (p. 11).

[R104] Villegas, N. M., Müller, H. A., & Tamura, G. (2011). Optimizing run-time SOA governance through context-driven SLAs and dynamic monitoring. In *International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)* (pp. 1–10). IEEE. https://doi.org/10.1109/MESOCA.2011.6049036

[R105] Wang, J., Yan, W. Q., Kankanhalli, M. S., Jain, R., & Reinders, M. J. T. (2003). Adaptive monitoring for video surveillance. In *Joint Conference of the 4th International Conference on Information, Communications and Signal Processing and 4th Pacific-Rim Conference on Multimedia (ICICS-PCM)* (Vol. 2, pp. 1139–1143). IEEE. https://doi.org/10.1109/ICICS.2003.1292638

[R106] Wang, M., Wang, H., & Xu, D. (2005). The design of intelligent workflow monitoring with agent technology. *Knowledge-Based Systems*, *18*(6), 257–66. https://doi.org/10.1016/j.knosys.2004.04.012

[R107] Wei, Y., & Blake, M. B. (2012). An agent-based services framework with adaptive monitoring in cloud environments. In *International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)* (pp. 4–9). IEEE. https://doi.org/10.1109/WETICE.2012.20

[R108] Willett, R., Martin, A., & Nowak, R. (2004). Backcasting: adaptive sampling for sensor networks. In *3rd International symposium on Information processing in sensor networks (IPSN)* (p. 124). New York, New York, USA: ACM Press. https://doi.org/10.1145/984622.984641

[R109] Zhang, W., & Cao, G. (2004). DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks. *IEEE Transactions on Wireless Communications*, *3*(5), 1689–1701. https://doi.org/10.1109/TWC.2004.833443

[R110] Zhou, J., & De Roure, D. (2007). FloodNet: Coupling adaptive sampling with energy aware routing in a flood warning system. *Journal of Computer Science and Technology*, *22*(1), 121–130. https://doi.org/10.1007/s11390-007-9017-7

A2 Data mining variables and results

Table A2-1
Variables used in Data Mining

Research sub-question	Id	Variable	Values
RQ2.1	v_1	Year *(of last approach contribution)*	2000-2016
RQ2.2	v_2	Type of publication *(of last approach contribution)*	Conference, Journal, Workshop
RQ2.3	v_3	Type of paper *(of last approach contribution)*	Industry, Academy
RQ2.4	v_4	Continent *(of last approach contribution)*	North America, South America, Europe, Asia, Oceania
RQ3.1	v_5	Type of contribution	Algorithm(s) and architecture, Algorithm(s)-only
RQ3.2	v_6	Solution generalizability	Problem-specific, Domain-specific, Generic
RQ4.1	v_7	Improve monitoring data characteristics	True, False
RQ4.1	v_8	Provide adaptation capabilities	True, False
RQ4.1	v_9	Reduce the impact of monitoring	True, False
RQ4.1	v_{10}	Respond to changes	True, False
RQ4.1	v_{11}	Satisfy systems' goals	True, False
RQ4.1	v_{12}	Solve a trade-off	True, False
RQ4.2	v_{13}	Metrics to monitor	True, False
RQ4.2	v_{14}	Monitoring operation	True, False
RQ4.2	v_{15}	Monitoring mechanism	True, False
RQ4.2	v_{16}	Monitoring system composition	True, False
RQ4.2	v_{17}	Sampling points	True, False
RQ4.2	v_{18}	Sampling rate	True, False
RQ4.3	v_{19}	Suspected problem	True, False
RQ4.3	v_{20}	SuM or monitoring system changes	True, False

RQ4.3	v_{21}	Monitored data characteristics	True, False
RQ4.3	v_{22}	Monitoring requirements changes	True, False
RQ4.3	v_{23}	Time	True, False
RQ4.3	v_{24}	Trigger open	True, False
RQ4.4	v_{25}	Algorithm	True, False
RQ4.4	v_{26}	Model-driven	True, False
RQ4.4	v_{27}	Analysis techniques not detailed	True, False
RQ4.4	v_{28}	Human analysis	True, False
RQ4.4	v_{29}	No analysis	True, False
RQ4.4	v_{30}	Probability/Statistics	True, False
RQ4.5	v_{31}	Human decision	True, False
RQ4.5	v_{32}	Objective function	True, False
RQ4.5	v_{33}	Policies	True, False
RQ4.5	v_{34}	Rules	True, False
RQ4.6	v_{35}	Manual	True, False
RQ4.6	v_{36}	Automatic	True, False
RQ4.6	v_{37}	Semi-automatic	True, False
RQ4.7	v_{38}	Parameter	True, False
RQ4.7	v_{39}	Structural	True, False
RQ5.1	v_{40}	Type of evaluation	Experiment, Industry use case, No evaluation
RQ5.2	v_{41}	Software applications	True, False
RQ5.2	v_{42}	Clouds/Grids	True, False
RQ5.2	v_{43}	Mobile sensors	True, False
RQ5.2	v_{44}	Network	True, False
RQ5.2	v_{45}	No evaluation	True, False
RQ5.2	v_{46}	Sensor networks	True, False
RQ5.2	v_{47}	Service/Component-based systems	True, False

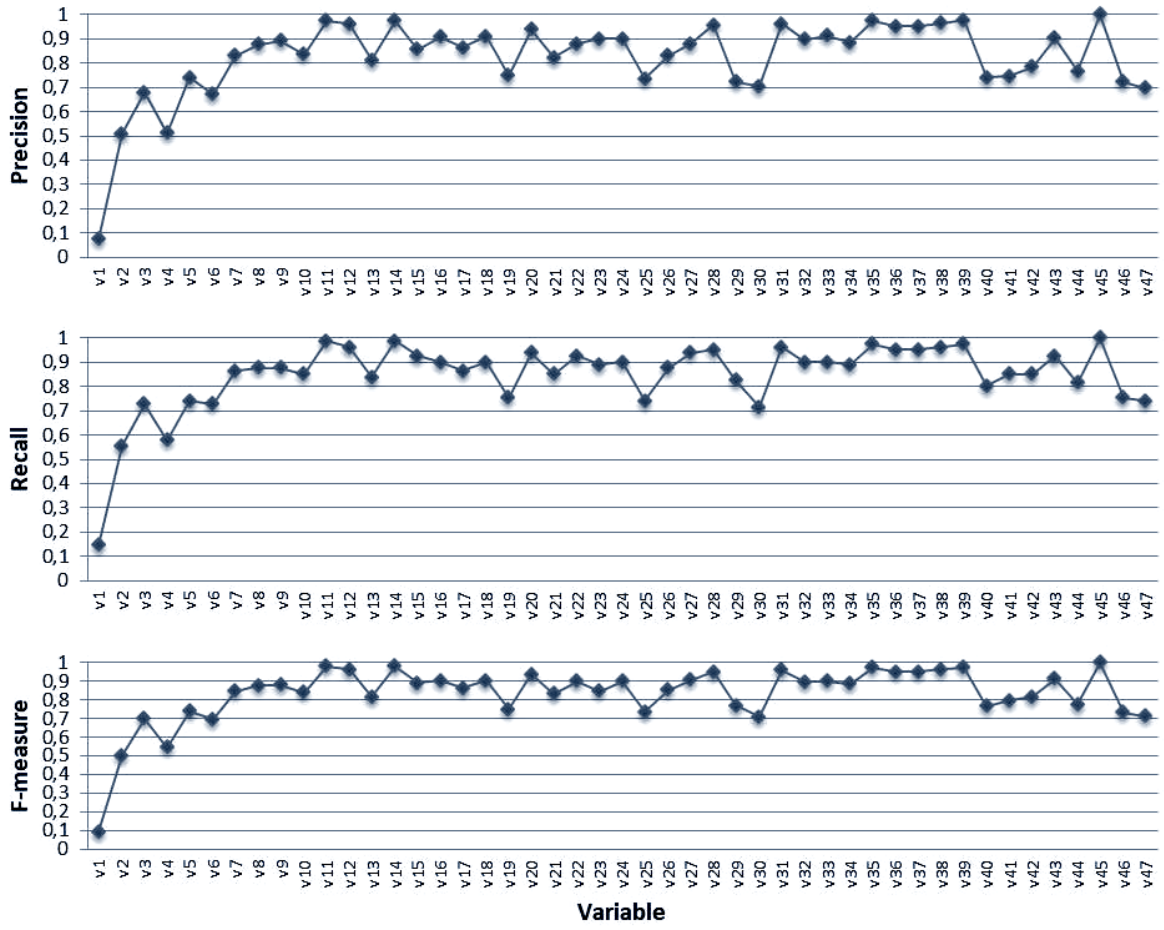


Figure A2-1: Resulting *precision*, *recall,* and *f-measure* per variable classifier

Table A2-2
Resulting relevant Data Mining classifiers

Variable	Rules	Interpretation
Satisfy systems' goals	*(for all)* Satisfy system's goals = False	In general, approaches are not motivated by the purpose of satisfying system's goals.
Solve a trade-off	*If* ((Reduce the impact of monitoring = True) or (Respond to changes = True) or (Improve monitoring data characteristics = True) or (Provide adaptation capabilities = True)) *then* (Solve a trade-off = False) *(True otherwise)*	Some adaptation purposes (reduce the impact of monitoring, respond to changes, improve monitoring data characteristics or provide adaptation capabilities) do not usually motivate an approach in conjunction with solving a trade-off purpose.
Monitoring operation	*(for all)* Monitoring operation = False	In general, approaches do not aim at adapting the monitoring operation.
Monitoring system	*If* (Structural = True and Sampling points = False and Suspected problem =	Structural changes executed on monitoring systems by approaches are

composition	False) ***then*** (Monitoring system composition = True) *(False otherwise)*	usually done for enacting monitoring system composition adaptation decisions, as long as they do not correspond to sampling points' adaptations and the adaptation trigger is not a suspected problem.
Sampling rate	***If*** (Structural = False and Sampling points = False) ***then*** (Sampling rate = True) *(False otherwise)*	Parameter changes are usually executed by approaches for adapting the sampling rate, except in the cases of non-structural sampling points' adaptation.
SuM or monitoring system changes	***If*** (Suspected problem = False and Trigger open = False and Monitored data characteristics = False and Time = False) ***then*** SuM or monitoring system change = True *(False otherwise)*	Approaches triggering adaptation by SuM or monitoring system changes do not tend to consider some kinds of triggers (suspected problem, open trigger, monitored data characteristics and time).
Trigger open	***If*** (Human analysis = True and SuM or monitoring system change = False) ***then*** (Trigger open = True) *(False otherwise)*	Approaches considering human analysis that do not trigger adaptations by SuM or monitoring systems changes, tend to leave the adaptation trigger open.
Human analysis	***If*** (Human decision = True) ***then*** (Human analysis = True) *(False otherwise)*	Approaches considering human-based decision-making usually also consider human-based analysis.
Human decision	***If*** (Human analysis = True or Manual = True) ***then*** (Human decision = True) *(False otherwise)*	Approaches considering human analysis or manual enactment of the adaptation decisions tend to conduct decision-making supported by humans.
Policies	***If*** (Objective function = True or Automatic = False or Rules = True) ***then*** (Policies = False) *(True otherwise)*	Policies are mainly used by existing approaches for making adaptation decisions, except for approaches that do not support automatic enactment or use objective functions or rules as decision-making criteria.
Manual	***If*** (Human decision = True and Semi-automatic = False) ***then*** (Manual = True) *(False otherwise)*	Approaches considering human-driven decision-making process tend to enact adaptations semi-automatically or manually.
Automatic	***If*** (Human analysis = True) ***then*** (Automatic = False) *(True otherwise)*	Most of the approaches considering human analysis do not consider automatic enactment.
Semi-automatic	***If*** (Automatic = False and Manual = False) ***then*** (Semi-automatic = True) *(False otherwise)*	Approaches supporting semi-automatic enactment do not support other kinds of enactment.

Parameter	*If* (Structural = False) *then* (Parameter = True) *(False otherwise)*	In general, approaches do not support the execution of both types of adaptation in a single solution.
Structural	*If* (Parameter = True and Monitoring system composition = False) *then* (Structural = False) *(True otherwise)*	
Mobile sensors	*If* (Objective function = True and Type of paper = Academy and Time = False) *then* (Mobile sensors = True) *(False otherwise)*	Approaches evaluated in mobile sensors systems do not trigger adaptations periodically and use objective functions for conducting their decision-making process. Moreover, most of them have been published by academics.
No evaluation	*If* (Type of evaluation= No evaluation) *then* (No evaluation= True) *(False otherwise)*	Approaches we have grouped in the *No evaluation* category in RQ5.1 were also correctly classified in RQ5.2 as not evaluated.

B

APPENDIX

How to improve: Study on SASs' self-improvement

B1 SASs' literature review references (identified in first manual search iteration)

- Cheng et al., 2009

[R1] Coulson, G. et al. (2008). A generic component model for building systems software. ACM Transactions on Computer Systems, 26(1), 1–42.

[R2] Fickas, S., & Feather, M. S. (1995). Requirements monitoring in dynamic environments. In Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95) (pp. 140–147). IEEE Comput. Soc. Press.

[R3] Finkelstein, A. (2008). Requirements reflection. Dagstuhl Presentation.

[R4] Kramer, J., & Magee, J. (2007). Self-Managed Systems: an Architectural Challenge. In Future of Software Engineering (FOSE '07) (pp. 259–268). IEEE.

[R5] Maes, P. (1987). Computional reflection (PhD thesis). Vrije Universiteit.

[R6] Robinson, W. N. (2003). Monitoring Web service requirements. In Proceedings of the IEEE International Conference on Requirements Engineering (Vol. 2003–Janua, pp. 65–74). IEEE Comput. Soc.

[R7] Robinson, W. N. (2006). A requirements monitoring framework for enterprise systems. Requirements Engineering, 11(1), 17–41.

[R8] Savor, T., & Seviora, R. E. (1997). An approach to automatic detection of software failures in real-time systems. In Real-Time Technology and Applications - Proceedings (pp. 136–146). IEEE Comput. Soc.

► De Lemos et al., 2013

[R9] Aßmann, U., Bencomo, N., Cheng, B. H. C., & France, R. B. (2012). Models@run.time (Dagstuhl Seminar 11481). Dagstuhl Reports 1(11), 91–123. https://doi.org/10.4230/dagrep.1.11.91

[R10] Bencomo, N., Blair, G., France, R., Muñoz, F., & Jeanneret, C. (2010). 4th International Workshop on Models@run.time (pp. 119–123). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-12261-3_12

[R11] Brun, Y., & Medvidovic, N. (2007). An Architectural Style for Solving Computationally Intensive Problems on Large Networks. In International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '07) (pp. 2–2). IEEE. https://doi.org/10.1109/SEAMS.2007.4

[R12] Georgiadis, I., Magee, J., & Kramer, J. (2002). Self-organising software architectures for distributed systems. In Proceedings of the first workshop on Self-healing systems - WOSS '02 (p. 33). New York, New York, USA: ACM Press. https://doi.org/10.1145/582128.582135

[R13] Goldsby, H. J., & Cheng, B. H. C. (2008). Automatically Generating Behavioral Models of Adaptive Systems to Address Uncertainty. In Model Driven Engineering Languages and Systems (pp. 568–583). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-87875-9_40

[R14] Malek, S., Mikic-Rakic, M., & Medvidovic, N. (2005). A decentralized redeployment algorithm for improving the availability of distributed systems. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 3798 LNCS, pp. 99–114). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11590712_8

[R15] Murray, R. M., Astrom, K. J., Boyd, S. P., Brockett, R. W., & Stein, G. (2003). Feature - Future directions in control in an information-rich word. IEEE Control Systems Magazine, 23(2), 20–33. https://doi.org/10.1109/MCS.2003.1188769

[R16] Sawyer, P., Bencomo, N., Whittle, J., Letier, E., & Finkelstein, A. (2010). Requirements-Aware Systems: A Research Agenda for RE for Self-adaptive Systems. In 2010 18th IEEE International Requirements Engineering Conference (pp. 95–103). IEEE. https://doi.org/10.1109/RE.2010.21

[R17] Vromant, P., Weyns, D., Malek, S., & Andersson, J. (2011). On interacting control loops in self-adaptive systems. In Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems - SEAMS '11 (p. 202). New York, New York, USA: ACM Press. https://doi.org/10.1145/1988008.1988037

[R18] Weyns, D., Malek, S., & Andersson, J. (2010). On decentralized self-adaptation. In Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '10 (pp. 84–93). New York, New York, USA: ACM Press. https://doi.org/10.1145/1808984.1808994

► Krupitzer et al., 2015

[R19] De Wolf, T., & Holvoet, T. (2005). Emergence versus self-organisation: Different concepts but promising when combined. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 3464 LNAI, pp. 1–15). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11494676_1

[R20] De Wolf, T., & Holvoet, T. (2003). Towards autonomic computing: Agent-based modelling, dynamical systems analysis, and decentralised control. In IEEE International Conference on Industrial Informatics (INDIN) (Vol. 2003–Janua, pp. 470–479). IEEE. https://doi.org/10.1109/INDIN.2003.1300381

[R21] De Wolf, T., & Holvoet, T. (2007). Design Patterns for Decentralised Coordination in Self-organising Emergent Systems (pp. 28–49). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-69868-5_3

[R22] De Wolf, T., & Holvoet, T. (2005). Towards a Methodology for Engineering Self-Organising Emergent Systems. Conference on Self-Organization and Autonomic Informatics. IOS Press. Retrieved from https://dl.acm.org/citation.cfm?id=1563536.1563540

[R23] Dowling, J., & Cahill, V. (2004). Self-managed decentralised systems using K-components and collaborative reinforcement learning. In Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems - WOSS '04 (pp. 39–43). New York, New York, USA: ACM Press. https://doi.org/10.1145/1075405.1075413

[R24] Krupitzer, C., Vansyckel, S., & Becker, C. (2013). FESAS: Towards a Framework for Engineering Self-Adaptive Systems. In 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (pp. 263–264). IEEE. https://doi.org/10.1109/SASO.2013.36

[R25] Nafz, F., Seebach, H., Steghöfer, J. P., Bäumler, S., & Reif, W. (2010). A formal framework for compositional verification of organic computing systems. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 6407 LNCS, pp. 17–31). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-16576-4_2

[R26] Tomforde, S., Prothmann, H., Rochner, F., Branke, J., Hähner, J., Müller-Schloer, C., & Schmeck, H. (2008). Decentralised Progressive Signal Systems for Organic Traffic Control. In 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems (pp. 413–422). IEEE. https://doi.org/10.1109/SASO.2008.31

[R27] Weyns, D. (2010). Architecture-based design of multi-agent systems. Architecture-Based Design of Multi-Agent Systems. Springer. https://doi.org/10.1007/978-3-642-01064-4

[R28] Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., ... Göschka, K. M. (2013). On patterns for decentralized control in self-adaptive systems. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 7475 LNCS, pp. 76–107). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35813-5_4

► Weyns, 2017

[R29] Bencomo, N., & Belaggoun, A. (2013). Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 7830 LNCS, pp. 221–236). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37422-7_16

[R30] Bojke, L., Claxton, K., Sculpher, M., & Palmer, S. (2009). Characterizing structural uncertainty in decision analytic models: A review and application of methods. Value in Health, 12(5), 739–749. https://doi.org/10.1111/j.1524-4733.2008.00502.x

[R31] Brun, Y., Di Marzo Serugendo, G., Gacek, C., Giese, H., Kienle, H., Litoiu, M., ...

Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 5525 LNCS, pp. 48–70). https://doi.org/10.1007/978-3-642-02161-9_3

[R32] Calinescu, R., Gerasimou, S., & Banks, A. (2015). Self-adaptive software with decentralised control loops. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 9033, pp. 235–251). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-46675-9_16

[R33] Iftikhar, M. U., & Weyns, D. (2014). ActivFORMS: active formal models for self-adaptation. In Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems - SEAMS 2014 (pp. 125–134). New York, New York, USA: ACM Press. https://doi.org/10.1145/2593929.2593944

B2 SASs' literature review references (identified in second manual search iteration)

[R1] Ali, R. (2010). Modeling and Reasoning about Contextual Requirements : Goal-based Framework (PhD Thesis). University of Trento.

[R2] Canavera, K. R., Esfahani, N., & Malek, S. (2012). Mining the execution history of a software system to infer the best time for its adaptation. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering - FSE '12 (p. 1). New York, New York, USA: ACM Press. https://doi.org/10.1145/2393596.2393616

[R3] Esfahani, N., Elkhodary, A., & Malek, S. (2013). A learning-based framework for engineering feature-oriented self-adaptive software systems. IEEE Transactions on Software Engineering, 39(11), 1467–1493. https://doi.org/10.1109/TSE.2013.37

[R4] Gullapalli, R., Muthusamy, C., & Babu, A. (2011). Data Mining in adaptive control of distributed computing system performance. Int. J. Comput. Trends Technol., 2(2), 128–133.

[R5] Inverardi, P., & Mori, M. (2011). Requirements models at run-time to support consistent system evolutions. In Proceedings of the 2011 2nd International Workshop on Requirements@Run.Time, RE@RunTime 2011 (pp. 1–8). IEEE. https://doi.org/10.1109/ReRunTime.2011.6046241

[R6] Knauss, A., Damian, D., Franch, X., Rook, A., Müller, H. A., & Thomo, A. (2016). Acon: A learning-based approach to deal with uncertainty in contextual requirements at runtime. Information and Software Technology, 70, 85–99. https://doi.org/10.1016/j.infsof.2015.10.001

[R7] Oriol, M., Qureshi, N. A., Franch, X., Perini, A., & Marco, J. (2012). Requirements monitoring for adaptive service-based applications. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 7195 LNCS, pp. 280–287). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-28714-5_25

[R8] Qureshi, N. A., & Perini, A. (2009). Engineering adaptive requirements. In Proceedings of the 2009 ICSE Workshop on Software Engineering for Adaptive and

Self-Managing Systems, SEAMS 2009 (pp. 126–131). IEEE. https://doi.org/10.1109/SEAMS.2009.5069081

[R9] Ramirez, A. J., Fredericks, E. M., Jensen, A. C., & Cheng, B. H. C. (2012). Automatically RELAXing a goal model to cope with uncertainty. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 7515 LNCS, pp. 198–212). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33119-0_15

[R10] Souza, V. E. S., Lapouchnian, A., & Mylopoulos, J. (2012). (Requirement) evolution requirements for adaptive systems. In ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (pp. 155–164). IEEE. https://doi.org/10.1109/SEAMS.2012.6224402

B3 SASs' literature review references (final set)

[R1] A. Knauss, D. Damian, X. Franch, A. Rook, H. A. Müller, and A. Thomo, "Acon: A learning-based approach to deal with uncertainty in contextual requirements at runtime," Inf. Softw. Technol., vol. 70, pp. 85–99, 2016.

[R2] I. Gerostathopoulos, D. Skoda, F. Plasil, T. Bures, and A. Knauss, "Architectural Homeostasis in Self-Adaptive Software-Intensive Cyber-Physical Systems," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7957, no. January, 2016, pp. 113–128.

[R3] V. Klos, T. Gothel, and S. Glesner, "Adaptive Knowledge Bases in Self-Adaptive System Design," Proc. - 41st Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2015, pp. 472–478, 2015.

[R4] D. Han, J. Xing, Q. Yang, J. Li, and H. Wang, "Handling Uncertainty in Self-Adaptive Software Using Self-Learning Fuzzy Neural Network," Proc. - Int. Comput. Softw. Appl. Conf., vol. 2, no. 1, pp. 540–545, 2016.

B4 SASs' self-improvement literature resources

Table B4-1
Journals

Id	Name
1	ACM Transactions on Intelligent Systems and Technology
2	ACM Transactions on Software Engineering and Methodology
3	Advances in Engineering Software
4	Computer
5	Decision Support Systems
6	Empirical Software Engineering
7	European Journal of Information Systems
8	IEEE Cloud Computing
9	IEEE Control Systems Magazine
10	IEEE Internet of Things Journal

11	IEEE Robotics & Automation Magazine
12	IEEE Software
13	IEEE Systems Journal
14	IEEE Transactions on Automatic Control
15	IEEE Transactions on Automation Science and Engineering
16	IEEE Transactions on Cloud Computing
17	IEEE Transactions on Control Systems Technology
18	IEEE Transactions on Dependable and Secure Computing
19	IEEE Transactions on Emerging Topics in Computing
20	IEEE Transactions on Mobile Computing
21	IEEE Transactions on Reliability
22	IEEE Transactions on Services Computing
23	IEEE Transactions on Software Engineering
24	Information and Software Technology
25	Information Systems Frontiers
26	International Journal of Robust and Nonlinear Control
27	ISA Transactions
28	Journal of Information Technology
29	Journal of Network and Computer Applications
30	Journal of Strategic Information Systems
31	Journal of Systems and Software
32	Journal of the Association for Information Systems
33	Pervasive and Mobile Computing
34	IEEE Transactions on Fuzzy Systems
35	IEEE Transactions on Evolutionary Computation
36	IEEE Transactions on Neural Networks and Learning Systems
37	IEEE Computational Intelligence Magazine
38	IEEE Transactions on Affective Computing
39	International Journal of Neural Systems
40	Knowledge-based Systems
41	Neural Computing & Applications
42	Applied Soft Computing
43	Swarm and Evolutionary Computation
44	Expert Systems with Applications
45	Integrated Computer-aided Engineering
46	Cognitive Computation
47	International Journal of Intelligent Systems
48	Advanced Engineering Informatics
49	Neurocomputing
50	Engineering Applications of Artificial Intelligence
51	IEEE Transactions on Knowledge and Data Engineering

Table B4-2
Conferences

Id	Name
1	Automated Software Engineering Conference
2	International Symposium on Software Reliability Engineering
3	International Symposium on Empirical Software Engineering and Measurement
4	International Conference on Web Information Systems Engineering
5	Americas Conference on Information Systems
6	International Conference on Advanced Information Systems Engineering
7	International Conference on Cooperative Information Systems
8	International Conference on Design Science Research in Information Systems and Technology
9	European Conference on Information Systems
10	International Conference on Information Systems Development
11	Pacific Asia Conference on Information Systems
12	European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering
13	International Conference on Software Engineering
14	International Conference on Information Systems
15	International Joint Conference on Autonomous Agents and Multiagent Systems
16	Fundamental Approaches to Software Engineering
17	Asia-Pacific Software Engineering Conference
18	International Conference on Evaluation of Novel Approaches to Software Engineering
19	Integration of Software Engineering and Agent Technology
20	International Symposium on Empirical Software Engineering
21	Euromicro Conference on Software Engineering and Advanced Applications
22	International Conference on Software Engineering and Formal Methods
23	International Conference on Information Systems Technology and its Application
24	International Conference on Knowledge-Based and Intelligent Information and Engineering Systems
25	IEEE International Conference on Autonomic Computing
26	International Conference on User Modelling, Adaptation, and Personalization

B5 SASs' self-improvement literature review references

[R1] Richard J. Anthony, Mariusz Pelc, and Witold Byrski. 2010. Context-aware Reconfiguration of Autonomic Managers in Real-time Control Applications. In Proceeding of the 7th international conference on Autonomic computing - ICAC '10 (ICAC '10), 73–74. DOI:https://doi.org/10.1145/1809049.1809061

[R2] Richard John Anthony. 2008. A versatile policy toolkit supporting run-time policy reconfiguration. Cluster Comput. 11, 3 (2008), 287–298. DOI:https://doi.org/10.1007/s10586-008-0058-7

[R3] Richard John Anthony. 2007. Policy-centric integration and dynamic composition of autonomic computing techniques. In ICAC. DOI:https://doi.org/10.1109/ICAC.2007.32

[R4] Richard Anthony, Dejiu Chen, Martin Törngren, Detlef Scholle, Martin Sanfridson, Achim Rettberg, Tahir Naseer, Magnus Persson, and Lei Feng. 2009. Autonomic

middleware for automotive embedded systems. In Autonomic Communication. 169–210. DOI:https://doi.org/10.1007/978-0-387-09753-4_7

[R5] Luciano Baresi and Liliana Pasquale. 2011. An eclipse plug-in to model system requirements and adaptation capabilities. In Proc. of the 6th IT-Eclipse Workshop. DOI:https://doi.org/33344444444

[R6] Luciano Baresi, Liliana Pasquale, and Paola Spoletini. 2010. Fuzzy goals for requirements-driven adaptation. In Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, RE2010, 125–134. DOI:https://doi.org/10.1109/RE.2010.25

[R7] Luciano Baresi and Clement Quinton. 2015. Dynamically Evolving the Structural Variability of Dynamic Software Product Lines. Proc. - 10th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst. SEAMS 2015 (2015), 57–63. DOI:https://doi.org/10.1109/SEAMS.2015.24

[R8] Jürgen Branke, Peter Goldate, and Holger Prothmann. 2007. Actuated traffic signal optimization using evolutionary algorithms. In Proceedings of the 6th European Congress and Exhibition on Intelligent Transport Systems and Services (ITS 2007). DOI:https://doi.org/10.1179/1743284713Y.0000000425

[R9] Jürgen Branke, Moez Mnif, Christian Müller-Schloer, Holger Prothmann, Urban Richter, Fabian Rochner, and Hartmut Schmeck. 2007. Organic Computing - Addressing complexity by controlled self-organization. Proc. - ISoLA 2006 2nd Int. Symp. Leveraging Appl. Form. Methods, Verif. Valid. (2007), 185–191. DOI:https://doi.org/10.1109/ISoLA.2006.19

[R10] Domenico Corapi, Daniel Sykes, Katsumi Inoue, and Alessandra Russo. 2011. Probabilistic rule learning in nonmonotonic domains. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 6814 LNAI, (2011), 243–258. DOI:https://doi.org/10.1007/978-3-642-22359-4_17

[R11] Christoph Dorn and Schahram Dustdar. 2010. Interaction-driven self-adaptation of service ensembles. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 6051 LNCS, (2010), 393–408. DOI:https://doi.org/10.1007/978-3-642-13094-6_31

[R12] Christoph Dorn, Daniel Schall, and Schahram Dustdar. 2009. Context-aware adaptive service mashups. 2009 IEEE Asia-Pacific Serv. Comput. Conf. APSCC 2009 c (2009), 301–306. DOI:https://doi.org/10.1109/APSCC.2009.5394107

[R13] A Elkhodary, N Esfahani, and S Malek. 2010. FUSION: A framework for engineering self-tuning self-adaptive software systems. In Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, 7–16. DOI:https://doi.org/10.1145/1882291.1882296

[R14] Ilenia Epifani, Carlo Ghezzi, Raffaela Mirandola, and Giordano Tamburrelli. 2009. Model Evolution by Run-Time Parameter Adaptation. In Proceedings - International Conference on Software Engineering, 111–121. DOI:https://doi.org/10.1109/ICSE.2009.5070513

[R15] Naeem Esfahani, Ahmed Elkhodary, and Sam Malek. 2013. A learning-based framework for engineering feature-oriented self-adaptive software systems. IEEE Trans. Softw. Eng. 39, 11 (November 2013), 1467–1493. DOI:https://doi.org/10.1109/TSE.2013.37

[R16] M. Usman Iftikhar and Danny Weyns. 2014. ActivFORMS: active formal models for self-adaptation. In Proceedings of the 9th International Symposium on Software

Engineering for Adaptive and Self-Managing Systems - SEAMS 2014, 125–134. DOI:https://doi.org/10.1145/2593929.2593944

[R17] M. Usman Iftikhar and Danny Weyns. 2014. Assuring system goals under uncertainty with active formal models of self-adaptation. In Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014 (ICSE Companion 2014), 604–605. DOI:https://doi.org/10.1145/2591062.2591137

[R18] Pooyan Jamshidi, Amir M. Sharifloo, Claus Pahl, Andreas Metzger, and Giovani Estrada. 2015. Self-Learning Cloud Controllers: Fuzzy Q-Learning for Knowledge Evolution. Proc. - 2015 Int. Conf. Cloud Auton. Comput. ICCAC 2015 (2015), 208–211. DOI:https://doi.org/10.1109/ICCAC.2015.35

[R19] Verena Klos, Thomas Gothel, and Sabine Glesner. 2015. Adaptive Knowledge Bases in Self-Adaptive System Design. Proc. - 41st Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2015 (2015), 472–478. DOI:https://doi.org/10.1109/SEAA.2015.48

[R20] Alessia Knauss, Daniela Damian, Xavier Franch, Angela Rook, Hausi A. Müller, and Alex Thomo. 2016. Acon: A learning-based approach to deal with uncertainty in contextual requirements at runtime. Inf. Softw. Technol. 70, (2016), 85–99. DOI:https://doi.org/10.1016/j.infsof.2015.10.001

[R21] Jeff Kramer and Jeff Magee. 2007. Self-managed systems: An architectural challenge. FoSE 2007 Futur. Softw. Eng. (May 2007), 259–268. DOI:https://doi.org/10.1109/FOSE.2007.19

[R22] Christian Krupitzer, Julian Otto, Felix Maximilian Roth, Alexander Frommgen, and Christian Becker. 2017. Adding Self-Improvement to an Autonomic Traffic Management System. In Proceedings - 2017 IEEE International Conference on Autonomic Computing, ICAC 2017, 209–214. DOI:https://doi.org/10.1109/ICAC.2017.16

[R23] Euijong Lee, Young-Gab Gab Kim, Young-Duk Duk Seo, Kwangsoo Seol, and Doo-Kwon Kwon Baik. 2018. RINGA: Design and verification of finite state machine for self-adaptive software at runtime. Inf. Softw. Technol. 93, September 2017 (2018), 200–222. DOI:https://doi.org/https://doi.org/10.1016/j.infsof.2017.09.008

[R24] Zoltan Adam Mann and Andreas Metzger. 2018. Auto-Adjusting Self-Adaptive Software Systems. In 2018 IEEE International Conference on Autonomic Computing (ICAC), 181–186. DOI:https://doi.org/10.1109/ICAC.2018.00030

[R25] Nenad Medvidovic, David S. Rosenblum, and Richard N. Taylor. 1999. A language and environment for architecture-based software development and evolution. Proc. 21st Int. Conf. Softw. Eng. - ICSE '99 (1999), 44–53. DOI:https://doi.org/10.1145/302405.302410

[R26] Hiroyuki Nakagawa, Akihiko Ohsuga, and Shinichi Honiden. 2012. Towards Dynamic Evolution of Self-Adaptive Systems Based on Dynamic Updating of Control Loops. 2012 IEEE Sixth Int. Conf. Self-Adaptive Self-Organizing Syst. (September 2012), 59–68. DOI:https://doi.org/10.1109/SASO.2012.17

[R27] Liliana Pasquale, Luciano Baresi, and Bashar Nuseibeh. 2011. Towards adaptive systems through requirements@runtime? In CEUR Workshop Proceedings, 13–24. DOI:https://doi.org/33344444444

[R28] Holger Prothmann, Fabian Rochner, Sven Tomforde, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck. 2008. Organic control of traffic lights. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial

Intelligence and Lecture Notes in Bioinformatics). Springer Berlin Heidelberg, Berlin, Heidelberg, 219–233. DOI:https://doi.org/10.1007/978-3-540-69295-9_19

[R29] Clément Quinton, Rick Rabiser, Michael Vierhauser, Paul Grünbacher, and Luciano Baresi. 2015. Evolution in dynamic software product lines: challenges and perspectives. Proc. 19th Int. Conf. Softw. Prod. Line - SPLC '15 (2015), 126–130. DOI:https://doi.org/10.1145/2791060.2791101

[R30] Fabian Rochner, Holger Prothmann, J. Branke, C. Müller-Schloer, and H. Schmeck. 2006. An organic architecture for traffic light controllers. Proc. Inform. 1, (2006), 120–127. Retrieved from http://subs.emis.de/LNI/Proceedings/Proceedings93/GI-Proceedings-93-19.pdf

[R31] Felix Maximilian Roth, Christian Krupitzer, and Christian Becker. 2015. Runtime evolution of the adaptation logic in self-adaptive systems. In Proceedings - IEEE International Conference on Autonomic Computing, ICAC 2015, 141–142. DOI:https://doi.org/10.1109/ICAC.2015.20

[R32] Amir Molzam Sharifloo, Andreas Metzger, Clément Quinton, Luciano Baresi, and Klaus Pohl. 2016. Learning and evolution in dynamic software product lines. In Proceedings of the 11th International Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '16, 158–164. DOI:https://doi.org/10.1145/2897053.2897058

[R33] D Sykes, W Heaven, J Magee, and J Kramer. 2008. From goals to components: A combined approach to self-management. SEAMS'08 Proc. 2008 Int. Work. Softw. Eng. Adapt. self-managing Syst. (2008), 1–8.

[R34] Daniel Sykes, Domenico Corapi, Jeff Magee, Jeff Kramer, Alessandra Russo, and Katsumi Inoue. 2013. Learning revised models for planning in adaptive systems. In 2013 35th International Conference on Software Engineering (ICSE), 63–71. DOI:https://doi.org/10.1109/ICSE.2013.6606552

[R35] Daniel Sykes, Jeff Magee, and Jeff Kramer. 2011. FlashMob: Distributed Adaptive Self-Assembly. In Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems - SEAMS '11, 100. DOI:https://doi.org/10.1145/1988008.1988023

[R36] Hossein Tajalli, Joshua Garcia, George Edwards, and Nenad Medvidovic. 2010. PLASMA: A plan-based layered architecture for software model-driven adaptation. In Proceedings of the IEEE/ACM international conference on Automated software engineering - ASE '10, 467. DOI:https://doi.org/10.1145/1858996.1859092

[R37] Gabriel Tamura, Norha M Villegas, Hausi A Muller, Laurence Duchien, and Lionel Seinturier. 2013. Improving context-awareness in self-adaptation using the DYNAMICO reference model. In 2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 153–162. DOI:https://doi.org/10.1109/SEAMS.2013.6595502

[R38] Sven Tomforde, Holger Prothmann, Fabian Rochner, Jürgen Branke, Jörg Hähner, Christian Müller-Schloer, and Hartmut Schmeck. 2008. Decentralised progressive signal systems for organic traffic control. In Proceedings - 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2008, 413–422. DOI:https://doi.org/10.1109/SASO.2008.31

[R39] N M Villegas, G Tamura, H A Müller, L Duchien, and R Casallas. 2013. DYNAMICO: A reference model for governing control objectives and context relevance in self-adaptive software systems. Lect. Notes Comput. Sci. (including Subser. Lect. Notes

Artif. Intell. Lect. Notes Bioinformatics) 7475 LNCS, (2013), 265–293. DOI:https://doi.org/10.1007/978-3-642-35813-5_11

[R40] Edith Zavala, Xavier Franch, Jordi Marco, Alessia Knauss, and Daniela Damian. 2015. SACRE: A tool for dealing with uncertainty in contextual requirements at runtime. In 23rd IEEE International Requirements Engineering Conference (RE), 278–279. DOI:https://doi.org/10.1109/RE.2015.7320437

[R41] Edith Zavala, Xavier Franch, Jordi Marco, Alessia Knauss, and Daniela Damian. 2018. SACRE: Supporting contextual requirements' adaptation in modern self-adaptive systems in the presence of uncertainty at runtime. Expert Syst. Appl. 98, (May 2018), 166–188. DOI:https://doi.org/10.1016/j.eswa.2018.01.00

[R42] Tianqi Zhao, Wei Zhang, Haiyan Zhao, and Zhi Jin. 2017. A Reinforcement Learning-Based Framework for the Generation and Evolution of Adaptation Rules. In Proceedings - 2017 IEEE International Conference on Autonomic Computing, ICAC 2017, 103–112. DOI:https://doi.org/10.1109/ICAC.2017.47

This thesis presents an architectural proposal to support adaptive feedback loops in self-adaptive systems, called HAFLoop (Highly Adaptive Feedback control Loop). HAFLoop extends the widely used MAPE-K loop, providing a generic structure for its elements, as well as the mechanisms required for coordinating their operation with their adaptation process. Given its importance, this thesis focuses on the adaptation of the Monitor element of the loop. The experiments, executed in the domain of smart vehicles, provide promising results both in simulation and in real environments.

EDITH ZAVALA

Thesis supervised by

Dr. Xavier Franch and Dr. Jordi Marco