Research Challenges on Multi-layer and Mixed-Initiative Monitoring and Adaptation for Service-Based Systems

Annapaola Marconi∗, Antonio Bucchiarone∗, Konstantinos Bratanis§, Antonio Brogi†,
Javier Cámara‡, Dimitris Dranidis§, Holger Giese¶, Raman Kazhamiakin∥,
Rogerio de Lemos∗∗, Clarissa Cassales Marquezan†† and Andreas Metzger††

∗SOA Unit, Fondazione Bruno Kessler
via Sommarive 18, 38123 Trento, Italy - Email: [marconi, bucchiarone]@fbk.eu
†Department of Computer Science, University of Pisa
Largo B. Pontecorvo 3, 56127 Pisa, Italy - Email: brogi@di.unipi.it
‡Departamento Engenharia Informática, University of Coimbra
R. Pedro Alpoim, Coimbra 3030-290, Portugal - Email: jcmoreno@dei.uc.pt
§South-East European Research Centre, CITY College, International Faculty of the University of Sheffield,
24 Proxenou Koromila Street, 54622, Thessaloniki, Greece - Email: [dranidis, kobratanis]@seerc.org
¶Hasso Plattner Institute, University of Potsdam
Prof. Dr. Helmert Strae 2-3, 14482 Potsdam, Germany - Email: holger.giese@hpi.uni-potsdam.de
∥SAYservice SRL
via alla Cascata 56C, Trento, Italy - Email: raman@sayservice.it
∗∗School of Computing, University of Kent
Canterbury, Kent, CT2 7NF, UK - Email: r.delemos@kent.ac.uk
††Paluno, University of Duisburg-Essen
Schützenbahn 70, 45117 Essen, Germany - Email: [Clarissa.Marquezan, andreas.metzger]@paluno.uni-due.de

Abstract—Adaptation of complex service-based systems is one of the most challenging research problems for the Future Internet. A considerable effort has been dedicated in recent years to address this problem. However, there are still several important issues that call for concrete solutions. In this paper, we present a set of research challenges for multi-layer and mixed-initiative adaptation and monitoring that may guide the research in this area for the next 5-10 years.

Keywords—Adaptation and Monitoring, Multi-layer Service-based Systems, User-centric Service-based Systems, Quality Assurance.

I. INTRODUCTION

Monitoring and adaptation (M&A) of Service-based Systems (SBSs) is one of the most challenging research problems for the Future Internet [1]. The service abstraction has become so pervasive that we are now building systems that are multi-layered in nature. Cloud-computing allows us to build software as a service on top of a dynamic infrastructure that is also provided as a service (IaaS). This complicates the development of self-adaptive systems because the layers are intrinsically dependent one of the other. Moreover, things get even more complex when we consider mixed-initiative SBSs, where the operational context is a mixed environment of people, services and run-time support systems. These applications, typical of the Internet of Services, must provide their functionality to potential users with the required/agreed qualities of services, cope with the unreliable network on which they operate, and also deal with the changes in their execution environment, in the partner services with which they interact, and in the users preferences and context. This means that the system needs to be able to detect such potential problems and adapt its behaviour respectively.

A considerable effort has been dedicated in recent years to address the problem of monitoring and adapting complex service-based systems. However, few proposed solutions consider the multi-layered nature of these applications (e.g. [2]–[4]), or the key role of the user context and of the execution environment for the operation and management of the system (e.g. [5]–[7]). In fact, there are still several important problems that call for concrete solutions.

In this paper, we present the research challenges for SBS monitoring and adaptation emerged during the S-Cube Research Roadmap Workshop (refer to [8] for an overview of the workshop results), attended by 40 experts in the field. The overall objective of the workshop was to identify challenges that may become relevant after and beyond the S-Cube project (in 5-10 years) and which have the potential to radically change existing thinking.

The rest of the paper is organized as follows: Section II gives an overview of recent research results on multi-layer and mixed-initiative M&A; Section III presents in detail the identified challenges for each topic; finally, Section IV presents the conclusions for this work.
II. RESEARCH RESULTS AND ACHIEVEMENTS

A significant effort has been dedicated in recent years to investigate the problem of adaptive service-based systems (SBSs). However, most existing M&A approaches address one specific functional layer at a time and do not consider the execution environment as a possible source of change. This makes them inadequate to be applied in real-world domains, where the context often plays a prominent role, and changes in one layer often affect others. If we do not consider the system as a whole, we can run into different kinds of misjudgements. For example, if we detect an unexpected behavior at the software layer we may be inclined to adapt at that same layer, even though a more cost-effective solution might be found either at the infrastructure layer, or by combining adaptations at both layers. Even worse, a purely software adaptation might turn out to be useless due to infrastructural constraints we fail to consider. Similar considerations are made in case of unexpected behavior at the infrastructure layer, or at both.

Solutions that do consider cross-layered systems tend to concentrate either on monitoring or adaptation. Concerning cross-layer monitoring, the work in [9] proposes an extensible framework for monitoring business, software, and infrastructure services, yet the approach does not support the correlation of terms monitored at different layers. The approach in [10] proposes a cross-layer monitoring approach that considers service and infrastructure level events, which are produced by services communicating via a distributed enterprise service bus. The approach does not correlate knowledge collected at the different levels.

Regarding cross-layer adaptation, the work in [11] presents an approach for adapting multiple applications that share common resources. However, they expect the users to perceive and model the conflicts manually. In [12] the authors propose a framework to support cross-layer self-adaptation in SBSs. They present a technologically agnostic middleware to support coordinated cross-layer adaptations by integrating interface and application layer adaptation mechanisms. In [13] the authors propose a framework for cross-layer adaptation of service-based systems comprised of organization, coordination and service layers. Within the S-Cube project several works investigated the cross-layer adaptation problem [14], [15]. The work in [14] analyzes the dependencies of key performance indicators (KPIs) on process quality factors from different functional levels of a SBS, and then an adaptation strategy is decided to improve all the negatively affected quality metrics in the SBS. In [15] the authors define an approach that introduces a cross-layer adaptation manager (CLAM). The approach relies on a cross-layer meta model of the SBS and a set of predefined domain specific rules to integrate and coordinate existing analysis and adaptation tools. Similarly to [15], the work in [16], [17] adopt a cross-layer representation of the application model. However, these works target a limited number of adaptation cases such as service replacement.

A key topic of investigation within the S-Cube project was the development of solutions supporting the whole adaptation life-cycle (i.e. monitor, analyze, plan, execute), while taking into account the cross-layer nature and the execution context of SBSs. This effort has produced important results [2]–[4] that significantly improved the state of the art. An approach that integrates cross-layer monitoring and adaptation of service-based systems is [2], where the authors propose a framework that integrates software and infrastructure specific monitoring and adaptation techniques, enabling cross-layer control loops in SBSs. All the steps in the control loop acknowledge the multi-faceted nature of the system, ensuring an holistic reasoning, and adapt the system in a coordinated fashion. In the developed prototype they focus on the monitoring and adaptation of BPEL processes that are deployed onto a dynamic infrastructure. Similarly, the work done in [3] proposes a solution to avoid SLA violations by applying cross-layer adaptation techniques. The novelty of the approach is the exploitation of all SBS layers for the prevention of Service Level Agreement (SLA) violations. The identification of adaptation needs is based on quality of service prediction, which uses assumptions on the characteristics of the running execution context. Multiple adaptation mechanisms are available to react on the adaptation need, acting on different layers of the SBS. The adaptation strategy chooses the right adaptation mechanism, coordinated by a multi-agent community. The framework developed in [4] addresses a different problem: how to model cross-layer SLA contracts. The proposed SLA contract model includes parameters of KPI, key goal indicators (KGI) and IT infrastructure type. The authors present a methodology for creating, monitoring, and adapting an SLA contract, in particular, leveraging aspects of Quality of Service (QoS) violations.

Concerning mixed-initiative M&A, few works have investigated the problem in depth and the results are still preliminary. The work in [6], [7] investigates adaptation of monitoring specifications through templates that dictate the generic rules parametric with respect to the context properties. Given the concrete context, the specific template that fits better the concrete environment is instantiated. In [5] different adaptation strategies are explicitly associated to the different context dimensions and properties, thus defining which forms of adaptation are allowed or preferred in case of different contextual changes.

III. CHALLENGES FOR MULTI-LAYER AND MIXED-INITIATIVE M&A

Despite the huge effort dedicated to the problem of multi-layer and mixed-initiative M&A, there are still several important problems that call for concrete solutions. In the following Sections we present the next important challenges emerged during the S-Cube Research Roadmap Workshop [8]. We have grouped these challenges into three
main topics: (i) adaptation in user-centric systems, (ii) functional and non-functional quality assurance of adaptive systems, and (iii) decentralized and multi-layer monitoring of system quality.

A. Adaptation in User-centric Systems

With the growth of the service market a more and more prominent role is gained by user-centric applications, where services are intended to be consumed directly by the end-user (e.g., personal agenda, electronic maps, on-line flight booking and check-in, restaurant finder, etc.), as opposed to business-centric (or B2B) services, which are used to provide business-to-business cooperation. In these applications, a user is equipped with the possibility to perform different actions realized as services or service compositions, that should be customized to accommodate the specific context of the user, being his physical context (e.g., location) preferences (e.g., his role), and operational settings (specific services available). Very often these applications are made available on top of mobile devices, thus bringing an additional infrastructural dimension to the SBS stack.

The distinguishing factor of the above type of applications refers to the fact that these applications operate in continuously changing environments. That is, the same application shall operate differently for different user roles with different preferences, and should run on different devices and consider different services available, etc. Indeed, these changes may require the adjustment of the application to better fit the new settings. But besides requiring the application to be context-aware and to adapt to changes in the context, the way the application is managed should also take into account different settings. In particular, the way the application is monitored and adapted (in case of failures, changing user requirements, etc.) should also follow the changes in the context in the SBS.

An important challenge that has been only partially addressed is user-driven monitoring: when the user context changes, the way the SBS is monitored may change as well, since the new settings may require to collect specific information or to monitor different elements that became part of it. To accomplish this it is necessary to be able to adapt the monitoring specification in reaction to context changes.

User-driven adaptation refers to the fact that same adaptation is made differently in different user contexts. User-centric services bring additional constraints to the adaptation problem. While in business-centric settings the services are orchestrated in order to accomplish a specific business task, user-centric service-based systems should allow the user to decide and control which tasks are executed and how. This requires the ability not only to automatically compose and adapt different, often unrelated, services on the fly, but also to generate a flexible interaction protocol that allows the user to control and coordinate composition execution. For example, the service selection performed when an SBS user deals with his personal activities (e.g., select cheaper hotel/flight services while organizing personal trip) is different from corporate activities (select more reliable and faster services). The issue here is to be able to identify proper adaptation strategies and capabilities when the SBS user context changes.

A key aspect of user-centric applications is the new role of the user. To effectively exploit the potential of the Future Internet, the users must become spectauthors: at the same time service consumers, service inventors, and data donors. The quality of service-based applications will depend on the types of inputs received by the users ranging from information about their profiles and preferences to their opinions about consumed services. An important challenge is concerned with the provision of ways that encourage and incentivise users to create and share contents and data with the system and the other peers in the network.

B. Functional and Non-functional Quality Assurance of Adaptive Systems

1) Preventing unwanted co-adaptation, malicious adaptations, as well as races and other anomalies: In service-oriented systems with evolution and self-adaptation multiple adaptation loops may co-exist that can influence each other if they affect directly or indirectly the overlapping parts of the system. This results in the problem that the adaptation loops may interfere and therefore the overall self-adaptation may not perform as intended. Two adaptation loops that interfere may try, for example, to steer the adaptation into opposite directions and thus result in an unstable adaptation behavior. Or due to some time-wise overlap of two adaptation loops the resulting changes may not be consistent as the latter adaptation cannot take the former into account as the monitoring input of the latter has been processed before the former effects the system. Also races are possible where the subtle timing influences the outcome of the overall adaptation in unwanted ways.

These loops are quite common since adaptation of an SBA usually aims at addressing various goals, such as (i) correcting faults contained in the SBA (aka. corrective adaptation), and (ii) adapting the SBA to new and yet unknown requirements (aka. perfective adaptation). When building adaptive SBAs that address two or more such goals, precautions must be taken to ensure that the interplay and the interactions between the different types of adaptations are considered, as otherwise this can lead to conflicting adaptations [18]. As an example, to address the goal of corrective adaptation, the service-based application might aim at replacing a failed service A with a service B, while at the same time the SBA, in order to address the aim of perfective adaptation, aims to replace service A with a service C, which promises to provide a better quality of service than service A. In fact, coordinating various adaptation goals is already considered one of the key challenges in self-adaptive software.
If we have conflicting adaptation goals we may even observe co-adaptation phenomena that work in the short run but fail in the long run as the adaptation has been too specific to the counterparts to be useful if the setting changes more radically. These problems already exist for closed systems planned beforehand. For open service-oriented systems with evolution and self-adaptation, multiple interfering adaptation loops may even come into existence that have not been planned and analyzed beforehand and thus the problems are even harder to tackle. The core challenge that has to be addressed that underlies the outlined problems can be summarized as follows: how can we ensure that the adaptation loops of an open service-oriented system that are assembled at run-time in an unplanned manner do not interfere in an unwanted manner?

The challenge and related solutions are yet rather unexplored. In control engineering usually only sufficiently independent closed control loops are employed and if necessary special means are used to decouple the loops (e.g., an inverse kinematic that allows to control orthogonal coordinates even though the actuators are not orthogonal is added) or the loops are joint at design time. Taking the more fuzzy nature of self-adaptation of software as well as the open nature of the considered systems into account, we can conclude that similar solution cannot work in our case. Therefore, to approach this problem space for open service-oriented systems, the following radical new directions are promising to be explored:

- Develop a theory and techniques for a decoupling (analogously to encapsulation in modular design) that can ensure at design-time that the considered adaptation loop cannot interfere with any other adaptation loop.
- Make adaptation loops first class citizens of the architecture to be aware of the problem and employ architectural patterns such as strict hierarchies with decoupled leaf elements or coordinated self-adaptation that ensure that interfering adaptation loops can be excluded at design-time.
- Use run-time models of the architecture, which includes the adaptation loops, to detect possible interfering of adaptation loops to exclude them at run-time.
- Develop a theory and techniques for an on-the-fly decoupling or compensation of the interference of adaptation loops and use run-time models of the architecture, which includes the adaptation loops, to detect possible interfering of adaptation loops to handle them with these techniques at run-time.

2) **Quality assurance techniques to prevent runtime design decisions / adaptations to lead to inconsistent situations:** Service-based systems increasingly have to become adaptive in order to operate and evolve in highly dynamic environments. Research on SBSs has already produced a range of adaptation techniques and strategies. However, adaptive SBSs are prone to specific failures that would not occur in static applications. Examples are: *faulty adaptation behaviour* due to changes not anticipated during design-time; *conflicts* that may arise if different layers of a SBS react to relevant changes by issuing contradictory adaptations concurrently; and *contradictory adaptations* that cannot be applied in combination since they would violate the requirements of the system or would not be able to solve the observed deviation from requirements. In certain situations the *timing of events* may be the source of another such kind of conflict. For example, it might happen that changes occurring on the infrastructure layer are so fast, that although they can be addressed efficiently on that layer, the impact they have on the composition layer leads to severe problems, as the execution of adaptation activities on the composition layer is not quick enough to follow. This may mean that new events continuously trigger new adaptations, leading to an adaptation stack overflow, possibly delaying other important adaptations or bringing the application to a halt completely.

We can have also situations where *oscillations* can occur. This happens when conflicting adaptations are issued in sequence. For example, the composition layer may receive an event that triggers an adaptation, which in turn leads to a critical change that leads to an adaptation on the infrastructure layer. It might happen that the adaptation of the infrastructure layer in turn leads to critical event on the composition layer, necessitating yet another adaptation. Obviously, if this undesired behavior is not observed and controlled, this can lead to a long series of mutual adaptations without ever reaching a stable system configuration, i.e., this can lead to an adaptation livelock. Finally, *race conditions* occur if the two layers apply adaptations and the final outcome depends on the order in which these adaptations are performed and completed. As an example, an adaptation on the composition layer may require the deployment of a new service on the cloud. Yet, if the adaptation of the cloud needed to provide additional resources for that new service is executed after the deployment of the new service, it might happen that the deployment will not succeed due to limited computational resources. Yet, if the resources would have been reserved before deployment, this adaptation would have been successful.

All the previous failure examples demonstrate that to have a SBS with its adaptation techniques implemented is not enough to guarantee its reliability and applicability in practice. We also need to ensure that each SBS adaptation is generated and executed correctly. For this, each adaptation approach together with the underlying adaptation toolkit and platform should provide dedicated techniques to be used to address the different kinds of adaptation failures discussed above. At the same time, the different failures pose new requirements, which may not be addressed with the existing techniques, and would need their evolution or even new approaches.

3) **Provision of assurances when architecting resilient service-oriented systems:** When architecting dependable service-oriented systems, the activities associated with
the provision of assurances are usually performed during development-time (e.g., service certification, formal verification of service compositions, etc.). However, when resilience (i.e., the persistence of dependability when the system, its environment, or its goals face changes [19]) is considered, the collection, structuring and analysis of evidence needs to be performed both at development-time and run-time. Some of the main challenges in providing assurances in the context of resilient service-oriented systems are:

- **Combining development-time rationale with run-time decision making.** While some sources of evidence for assurances, such as the signature or protocol descriptions of services, can be used to provide some guarantees about the system at development-time (e.g., with respect to behavioural service compatibility), other sources of evidence that play a key role in maintaining resilience throughout the system’s execution (e.g., service availability) are only available through direct monitoring of the system at run-time. This dichotomy demands both appropriate scoping and flexible combination of the different sources of evidence to enable the coordination and effective enactment of run-time adaptation mechanisms.

- **Selecting and deploying during run-time the appropriate verification and validation tools and techniques for the generation of evidence.** One important aspect of service-oriented systems is the different degrees of persistence of service bindings. From completely fixed structures that implement replication of instances of the same services to improve dependability, to systems that are able to discover and bind services at run-time (e.g., replacing a service that fails to respond with a different, but functionally equivalent one), each one of them requires a different approach to verification and validation to provide assurances specific to system goals. The variety of techniques and tools that can be used in each case can range from model-checking and well-established testing techniques (e.g., for functionality, interoperability, security), to run-time verification or online testing [20], which are required in cases where service bindings are transient and performed at run-time.

- **Analyzing the collected evidence in order to build arguments that should be evaluated against the goals of the system.** The decision of whether to deploy or not a particular run-time adaptation of the system should be performed autonomously during run-time depending on the system operational boundaries. Since the analysis of evidence for building assurance arguments is a demanding process, it should be restricted to situations in which there is a clear evolution of the system (i.e., representative changes, such as a change in system goals), in contrast to changes of a lesser impact for which simpler resorts may suffice.

Another long term challenge regarding assurances as the system evolves is related to the need for **dynamically generating processes for managing the provision of assurances** (i.e., collection, structure and analysis of evidence) before the deployment of new versions of the system [21], considering the fact that the system may require different degrees of assurance as it adapts in reaction to changes.

4) **Concepts and techniques for formally guaranteeing the correctness of adaptations:** Software adaptation remains one of the crucial issues in system integration. Software adaptation is needed in a number of situations, ranging from the need of overcoming various types of mismatches among applications developed by different parties, of customizing services to different types of clients, of adapting legacy systems to meet new business demands, or of ensuring backward compatibility of new service versions.

Many approaches to software adaptation have been proposed in the literature. They differ for the type of addressed mismatches (e.g., signature, protocol, QoS, security), for the type of featured adaptation (static vs. dynamic, semi-automated vs. automated, single-layer vs. cross-layer), and for the type of implementation techniques employed to achieve the adaptation (e.g., connectors, adapters, aspect-oriented programming).

We argue that, in spite of the large body of activity that has been devoted to software adaptation, only few proposals have tried to address the fundamental issue of the correctness of the devised adaptation (e.g., [22], [23]). Roughly speaking, while developers can nowadays exploit various adaptation techniques to solve the adaptation needs of their systems, no guarantee on the behaviour of the adapted system is generally provided. In contrast, having some guarantee on the effects of the enforced adaptation is of paramount importance especially when adaptation may affect protocol or security behaviour of a system, or when adaptation affects multiple layers of the system.

The **application of formal methods to rigorously establish the correctness of adaptation techniques** can therefore be singled out as one of the important, largely open challenges in software adaptation. A related open challenge is to **formally establish the completeness of adaptation techniques**, that is, to rigorously characterise the set of mismatches that can be solved by applying a given adaptation technique.

C. **Multi-layer and Decentralized Monitoring**

1) **Decentralized Monitoring of System Quality:** Most of current monitoring and prediction techniques used by service integrators (or service brokers) consider information retrieved from centralized points, such as BPEL engines or application servers, associated with the SOA layers (Business Process Management, Service Composition, Service Infrastructure). Initially, this information from the different layers was analyzed in an isolated fashion. Nowadays, more sophisticated monitoring and prediction techniques are able to use cross-layer information to support the adaptation...
decisions in service-oriented systems. Nevertheless, the information used in cross-layer monitoring and prediction techniques is restricted to the SOA layers. The main problem of this restriction is the fact that detailed information from a major contributor for the need of adaptive service-oriented systems is left aside: the information from the network infrastructure.

The use of response time as a common metric from monitoring and prediction techniques is one example that clarifies the need for more sophisticated cross-layer techniques also able to consider network aspects. Depending on the observer point of view, i.e., end user, service integrator, service provider, the complexity of monitoring or predicting the response time can be completely different. In the case of predicting the end-to-end QoS of a service-oriented system (considering the end user perspective), the response time would include: processing time and network latency inside each service provider and service integration infrastructures, and network latency (e.g., within the Internet) among service provider, service integrator, and end user. This way, techniques to combine information from SOA layers with the one coming from the network infrastructure might enable more accurate results in the prediction models which allow better adaptive actions in service-oriented systems. In addition, the use of information from the communication networks is not only relevant for determining the response time QoS attribute from service-oriented systems, but it can also be used to better understand the reliability of such systems.

One important issue that rises under the scope of monitoring and predicting the whole service delivery chain (i.e., cross-layer including communication networks) is the use of centralized or decentralized strategies for the monitoring and predicting models. Centralized models, like the current ones, will not be enough in such scope because either they are not able to access information from other entities involved in the service-oriented system (e.g., service integrator has no access to information from service provider) or the centralized entities will become a bottleneck of the system (e.g., the service integrator cannot be responsible for gathering alone extra information about all end user network connectivity conditions, it would not scale). The problem of centralized solutions tends to be even stronger with the increasing use of thirdparty services, including Software-as-a-Service (i.e., applications for users), software services (i.e., software components for application developers and system integrators), as well as physical and human-provided business services (e.g., in SmartCities the citizens will be able to use services provided by IoT and also to provide services themselves from their mobile device, for instance).

2) Cross-layer Correlation of Observations, Predictions, and Events: A key component for implementing cross-layer monitoring and adaptation (M&A) in a service-based system is the correlation of monitored information from all layers. So far existing approaches to multi-layer M&A [2], [10], [12], [13], [15] have only leveraged correlation of information, mainly low level (basic) events observed during monitoring, for inferring higher level (complex) events using predefined static correlation rules. Existing literature related to multi-layer M&A has demonstrated in a limited scope how the correlation of information can be used. However, there are several challenges that remain unaddressed.

A challenge associated with the correlation of useful information in multi-layer M&A is the correlation of observations, predictions, and events from different sources, across the functional layers, and provided by different analysis, decision and adaptation mechanisms. For instance, monitored events correlated with predictions concerning future situations, decisions about adaptation strategies, and results of adaptation actions. The correlation of information produced not only by the monitoring mechanisms but also by analysis, decision, and adaptation mechanisms, will potentially result in more fruitful conclusions regarding the current situation. Also, the correlation of similar information from various sources (e.g., similar monitors) is required to confirm an observation or an inferred conclusion. The information inferred through correlations can be used for populating an internal knowledge model concerning various aspects of the SBS. The knowledge model could be leveraged for analysis and decision in multi-layer M&A.

Directions for investigating the aforementioned challenge comprise the horizontal correlation, i.e., to correlate observations, predictions, and events provided by different mechanisms operating in the same layer, the vertical correlation, i.e., to correlate information propagated across the functional layers of the SBS, establishment of common semantics for mapping the information provided by existing monitor, analysis, decision, adaptation techniques, and abstracting away the correlation logic from the concrete mechanisms.

Finally, another dimension of the challenge is the significant effort and expert knowledge that is required to define correlation rules at design-time. The problem becomes more complex in the case of multi-layer M&A, because there are more information sources to consider during correlation. Therefore, another challenge associated with the correlation of information in multi-layer M&A is the dynamic correlation of information. That is, to learn correlation rules automatically, such that the SBS makes correlations, which were not foreseen. The learning of correlation rules could potentially contribute to the overall ability of the SBS to adapt to unforeseen situations.

IV. Conclusion

Despite the huge effort dedicated in recent years to the problem of multi-layer and mixed-initiative monitoring and adaptation of SBSs, there are still several important problems that call for concrete solutions.

In this paper we presented the challenges emerged during the S-Cube Research Roadmap Workshop [8] attended by 40 experts in the field. The identified challenges cover three
main aspects (adaptation in user-centric systems, quality-assurance of adaptive systems, multi-layer and decentralized monitoring) and may guide the research in this area for the next 5-10 years.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreements 215483 (S-Cube).

REFERENCES


