МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2016
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2016
*Proceedings of the Forty Fifth Spring Conference
of the Union of Bulgarian Mathematicians
Pleven, April 6–10, 2016*

SELF-ADAPTATION IN SERVICE ORIENTED SYSTEMS[*]

Krasimir Baylov, Aleksandar Dimov

Due to increasing complexity of enterprise systems their support and evolution become an extremely difficult task. This way, such systems require research in mechanisms that allow them to change their behavior at runtime and self-adapt based on the changes in the surrounding environment. Although, some efforts in the area already exist, we do not have a sound, systematic and universally accepted approach for self-adaptation of service based software systems. The paper presents a study on self-adaptability mechanisms in software systems, analyzes them for applicability in service oriented architectures and provides a comparative analysis of such mechanisms and points the particular levels of abstraction at which they can be achieved.

1. Introduction. Service Oriented Architecture (SOA) plays a key role in enabling integration between information systems within enterprises that are otherwise incompatible. It provides different means for reuse, growth and interoperability and allows data exchange via web services [1]. Clients can search for web services in registries and obtain additional information regarding their functionality, providers, versions, etc. Despite the many benefits that SOA provides, managing their complexity is a major problem of contemporary systems [2, 3]. They become so complex that people could hardly maintain and evolve them. Authors of [4] argue that the only option to solve this is to implement *autonomic computing*. Autonomous systems can react to changes in the environment they work in and adapt based on initially defined high level goals.

Matthias et al. [7] provide an industrial case study regarding the constraints on designing variability-intensive service-oriented reference architectures. They put the focus on their study on designing systems that could be deployed in different context with minimal design and code changes by analyzing the constraints that need to be considered. The report identifies ten constraints but they should be more strictly considered in design phase. The problem presented in this case study clearly reveals the need and benefits of designing self-adaptive service oriented architectures.

This paper presents an ongoing research on applicability of self-adaptive aspects in service oriented environments. We believe that a new layer should be introduced that can support self-adaptability of SOA systems. Such a layer should be based on existing referent architectures, so that it could easily be applied on top of applications that are

[*]**2010 Mathematics Subject Classification:** 68N99.

Key words: service oriented architecture, software services, self-adaptive systems.

Research, presented in this paper was partially supported by the DFNI I02-2/2014 project, funded by the National Science Fund, Ministry of Education and Science in Bulgaria.

already designed and deployed in productive environment. We provide an overview of the key layers of service oriented systems and the different associated aspects that relate to their self-adaptive capabilities. As a result of the current study we introduce a new layer in service oriented architectures that is necessary in order to provide self-adaptive mechanisms in service based software systems. For this purpose, we have performed a detailed study on the modelling dimensions, properties and mechanisms used to reason about self-adaptive systems in general and afterwards, we have studied SOA reference models and various levels of abstraction at which they are designed.

The rest of the paper is structured as follows. Section 2 presents the self-adaptability aspects of information systems; Section 3 provides an overview of SOA reference architectures and discusses the mechanisms for building self-adaptive service oriented systems by proposing a new layer supporting self-adaptation and finally, Section 4 concludes the paper and points the directions of future research

2. Aspects of self-adaptability in information systems. The following section presents some of the key aspects related to self-adaptive systems. We have selected the ones are directly related to the self-adaptive capabilities of an enterprise software system.

2.1. Control loop. Self-adaptive systems can dynamically change their behavior or properties based on external stimulus. A key mechanism used for running the self-adaptation is the so called *control loop* or *feedback loop* [3]. It contains four main phases shown in Figure 1.

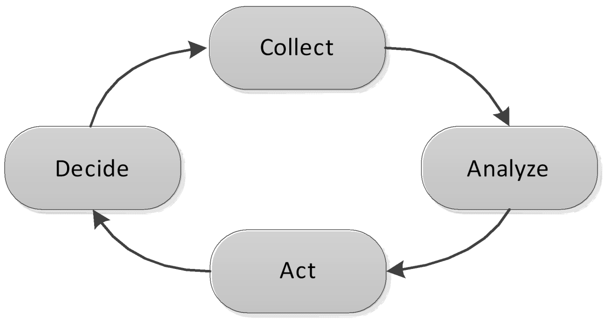

Fig. 1. Control Loop in Self-Adaptive Systems

In the *collect* phase, the relevant data is obtained from the surrounding environment and stored for further processing. Usually, this data is collected through agents or sensors. Then this data is *analyzed* and related to existing models. This lets the *decide* phase select the best action from a list of options. Finally, in the *act* phase, the system puts the change in place and the loop starts over.

2.2. Self-* properties. Self-adaptive systems can be viewed from multiple points depending on their direction of autonomy. These points are called *self-* properties* of the system. They represent the ability of the system to take autonomous actions in the following 4 areas [4]:

- **Self-configuration** – the system is able to configure itself in order to comply with initially defined high level goals
- **Self-optimization** – the systems is able to detect and optimize weak points or

places that can be improved in terms or performance, availability, etc.

- **Self-healing** – the system is able to detect problems, create a strategy for fixing them and applying the fix
- **Self-protection** – the systems is able to detect potential threats and defend itself against external attacks.

Although, the four properties of self-adaptability can be implemented independently, modern software systems would merge them in a single general property – *self-maintenance* [4]. This means that contemporary autonomous systems would have all of the four properties to achieve a greater degree of self-adaptability.

2.3. Modelling dimensions. Modeling dimensions provide different perspectives to reason about self-adaptive systems. They represent the points of variations to which a system can be considered self-adaptive [8]. There are four widely recognized groups of modelling dimensions:

- **Goals** – goals represent the objectives that a system should achieve
- **Change** – changes represent the cause that has triggered the adaptation of the system
- **Mechanisms** – mechanisms represent the reaction of the system towards the change
- **Effects** – effects represent the impact that the adaptation has over the system

Each of the described groups has a number of dimensions that provide a more detailed representation. They are presented in Figure 2.

Each of the presented dimensions provides system aspects that should be considered during design of self-adaptive systems. For example, systems goals can be viewed in terms of their ability to evolve over time (evolution), how rigorous they are (flexibility), the number of goals that the systems tries to follow (multiplicity), etc.

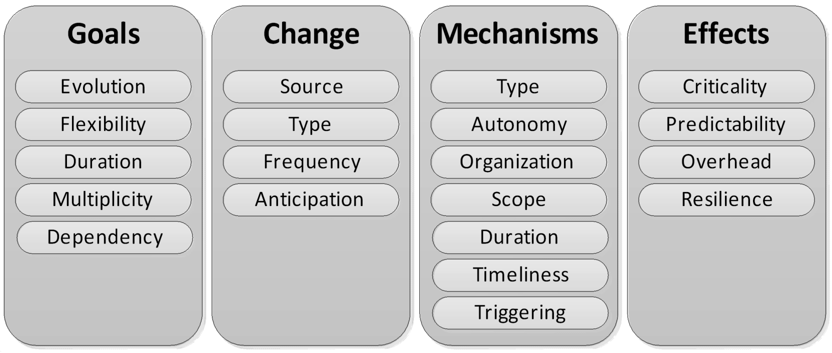

Fig. 2. Modelling dimensions for self-adaptive systems

Changes represent the stimulus that causes the system to self-adapt. When designing self-adaptive systems architects should consider what are the sources of the change (internal or external), the type of change (functional or non-functional), how often changes happen and can the change itself be foreseen.

Mechanisms provide design considerations on the type of change implementation (parametrical or structural), the level of autonomy when applying the change (assisted or autonomous), time that it will need in order to complete.

Effects refer to the final result of the adaptation. It can be viewed in terms of criticality (harmless to mission-critical), predictability (non-deterministic to deterministic), overhead (insignificant to failure) and resilience (resilient to vulnerable).

All of the presented modelling dimensions play a key role in the design of autonomous software systems. They should be considered carefully and the relevant trade-offs should be analyzed in order to design a high quality self-adaptive systems.

3. Self-Adaptive SOA. The following section provides an overview of SOA modeling principles. An overview of SOA modelling layers is presented and used as a ground for our research of analyzing the possibilities for self-adaptability in service oriented systems.

3.1. SOA modelling layers. There exist many and sometimes conflicting definitions of SOA [1]. However, in this paper, we refer to SOA as an *architectural style that emphasizes implementation of components as modular services* [6]. As such, SOA provides great flexibility in designing and building enterprise systems. However, big systems are built using layered styles in order to reduce the overall complexity. Ali.A [5] describes SOA as a layered architecture. Figure 3 presents the layers of SOA.

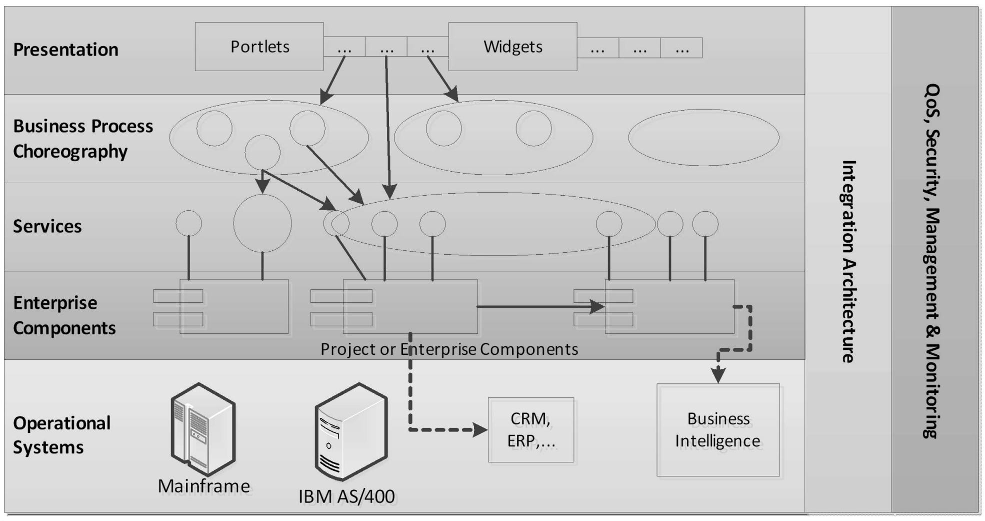

Fig. 3. SOA layers

This layered presentation of SOA means that self-adaptability also could be applied at different layers. For example, a given business process that runs on top of the services layer may not satisfy the initial requirements (functional, non-functional, etc.). In this case, the business process needs to self-adapt in order to comply with them. However, changes can be applied at various levels.

- **Business process** – the business process can be updated by modifying the workflow and by this, improve the overall efficiency. Sometimes a single web service could be replaced with another that provides better quality characteristics.
- **Services** – a single web service can be updated through a self-adaptation mechanism. If the service fails with a big number or requests a self-healing approach could be

applied in order to improve its reliability, availability, etc.

- **Enterprise components** – an enterprise component can be either updated or used to trigger and implement some self-adaptive actions. For example, a load balancer could redirect requests to a service instance that performs better due to geographic location deployment.
- **Operational system** – the operational system can be configured in order to satisfy the requirements for some of the upper levels. For example, a performance problem, detected at the business process level could be solved by applying small changes at this level like updating the network protocol, updating the hosts file, etc.
- **Integration (ESB)** – this layer enables the services integration through a set of capabilities like content based routing, data transformation, etc. They could be self-configured in order to satisfy the high level goals of the system.
- **Presentation** – in certain situations the presentation layer may require some runtime modifications like changing the color scheme or layout of the UI in order to make it more intuitive to the user.

The layered view of service oriented architectures, presented in Figure 3 provides one more layer named "*QoS, Security, Management & Monitoring*". We would classify it as a supporting layer. It does not provide any functional implementation or quality of service (QoS) support that are directly related to system behavior. However, this layer is responsible of monitoring the entire system, collect performance data and make it available for further processing. This is a key aspect of self-adaptive systems because they cannot make any analysis or decision without having data that represents the system behavior.

3.2. Designing self-adaptive SOA. Design of self-adaptive SOA systems should consider the self-adaptive aspects presented in section 2. A crucial part of the modeling is deciding the layers at which the self-adaptation mechanisms will be introduced. For example, a SOA systems, that is based on the layers presented on Figure 3 may adapt either at a single layer like *Business Process Choreography* or at multiple layers. Single layer adaptations provide limited scope adaptability. For example, performance problems detected at the *Business Process Choreography* layer can be resolved by replacing the web services or optimizing the workflow. However, such problems can better be resolved at networking or service implementation level, which are outside the current scope.

A challenge with the multiple adaptation layers is the synchronization between the separate layers. Adaptations may depend or may not depend on each other at runtime. In case they need to coordinate their decisions and actions, a communication protocol should be established or some middleware for self-adaptation should be used. In this case the *Architecture Integration Layer* (ESB) needs to act as a coordinator in the *analysis* and *decide* phase of the control loop.

Based on the different layers for applying self-adaptation for service oriented systems, we provide an updated version for SOA layering in terms or self-adaptation. We introduce a new adaptation layer that is responsible for the self-management aspects of a service oriented system in Figure 4. The new adaptation layer is placed across all vertical layers as it provides self-adaptive capabilities to all of them. In addition, this layer communicates directly with the *Integration Architecture* layer to provide support in synchronizing the actions when applying more complex adaptation strategies.

The control loop can be applied at all layers in order to guarantee self-adaptation at

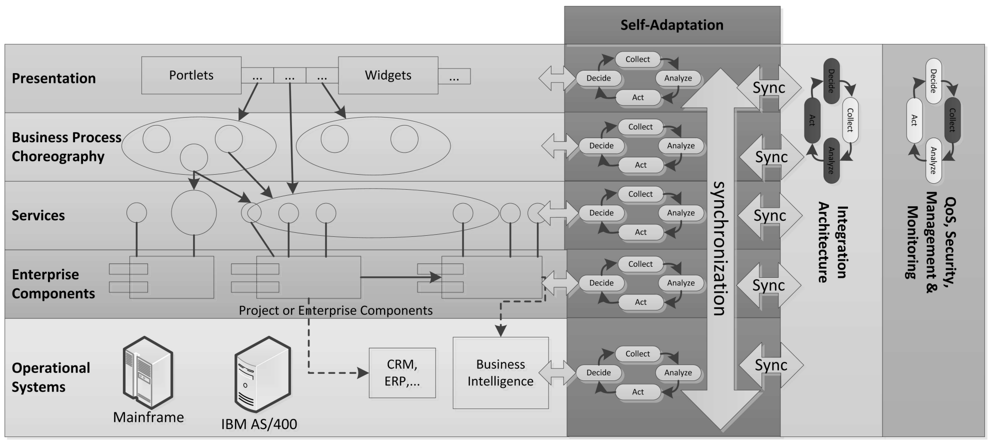

Fig. 4. Self-adaptive SOA layers

each of them. The control loop is also a required part for the *Integration Architecture* and *QoS Monitoring* layers but it may not be fully functional. For example in the *QoS Monitoring* layer the collect operation is the most important one since this is the main work that this layer is responsible for. However, phases like analyze, decide and act are not primary.

The provided enhanced SOA layered architecture is a result of an ongoing research. Some changes are likely to be applied as we analyze additional systems and adaptation approaches for service oriented systems.

4. Conclusions. This paper presented the results of an ongoing research in analyzing the self-adaptive aspects of service oriented architectures. The main self-adaptation aspects were summarized – control loops, self-* properties and modeling dimensions. As a foundation for this research we have used a layered SOA architecture and analyzed the levels at which self-adaptive mechanisms can be applied. We conclude that a SOA system can be adapted at various levels.

Our main contribution is a new layer for self-adaptation that allows the system to respond to internal and external changes with minimal or no human supervision. This approach is based on a standard SOA layered model and doesn't require designing the system from scratch. In other words, self-adaptive functionality can be added on top of existing service oriented systems.

There are many aspects that still need to be studied and analyzed for applicability in service oriented architectures. Additional aspects that need to be analyzed are – *requirements, engineering, assurances, design space, processes, decentralization of control loops*, etc. [9][10]. These aspects are generic to self-adaptive systems. As we analyze some of the SOA specifics, additional aspect may be identified which are specific for service oriented systems.

Although the current research in based on studying existing papers and systems, a simulation model or a prototype could be built to verify the conclusions made. Such a system can be used for developing self-adaptive agents that can be reused in other existing systems.

REFERENCES

[1] C. M. MacKenzie et al. Reference model for service oriented architecture 1.0. OASIS Standard 12, 2006.

[2] A. G. Ganek, T. A. Corbi. The dawning of the autonomic computing era. *IBM Syst. J.* **42**, No 1 (2003), 5–18.

[3] Y. Brun, G. M. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, M. Shaw. Engineering Self-Adaptive Systems through Feedback Loops. In: Software Engineering for Self-Adaptive Systems (eds B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee) Lecture Notes in Computer Science, Vol. **5525**. Springer-Verlag, Berlin, Heidelberg, 2009, 48–70

[4] J. O. Kephart, D. M. Chess. The vision of autonomic computing. *Computer*, **36**, No 1 (2003), 41–50.

[5] A. Arsanjani. Service-oriented modeling and architecture. IBM developer works, 2004, 1–15.

[6] L. Srinivasan, J. Treadwell. An overview of service-oriented architecture, web services and grid computing. HP Software Global Business Unit 2, 2005.

[7] M. Galster, P. Avgeriou, D. Tofan. Constraints for the design of variability-intensive service-oriented reference architectures – an industrial case study. *Inf. Softw. Technol.*, **55**, No 2 (2013), 428–441.

[8] J. Andersson, R. Lemos, S. Malek, D. Weyns. Modeling Dimensions of Self-Adaptive Software Systems. In Software Engineering for Self-Adaptive Systems (eds B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee). Lecture Notes in Computer Science, Vol. **5525**, Springer-Verlag, Berlin, Heidelberg, 2009, 27–47.

[9] B. H. C. Cheng et al. Software engineering for self-adaptive systems: A research roadmap. Software engineering for self-adaptive systems. Springer Berlin Heidelberg, 2009, 1–26.

[10] R. De Lemos et al. Software engineering for self-adaptive systems: A second research roadmap. Software Engineering for Self-Adaptive Systems II. Springer Berlin Heidelberg, 2013, 1–32.

Krasimir Baylov
Department of Software Engineering
Faculty of Mathematics and Infomratics
University of Sofia St. Kl. Ohridski
5, J. Bourchier Blvd
1164 Sofia, Bulgaria
e-mail: krasimirb@fmi.uni-sofia.bg

Aleksandar Dimov
Department of Software Engineering
Faculty of Mathematics and Infomratics
University of Sofia St. Kl. Ohridski
5, J. Bourchier Blvd
1164 Sofia, Bulgaria
e-mail: aldi@fmi.uni-sofia.bg
and
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8
1113, Sofia, Bulgaria

САМОАДАПТАЦИЯ В СОФТУЕРНИТЕ СИСТЕМИ ОРИЕНТИРАНИ КЪМ УСЛУГИ

Красимир Байлов, Александър Димов

Поддръжката и развитието на съвременните софтуерни системи, които се използват в големите корпорации, стават все по трудни, вследствие на увеличаващата се сложност на тези системи. Поради тази причина са необходими изследвания в областта на механизмите за възможност за промяна на системите, в следствие на съответните промени на контекста, в който те се изпълняват. Промените трябва да стават самостоятелно, без човешка намеса и по време на изпълнение, т.е. – чрез самоадаптация. В същото време, повечето съвременни системи се разработват според принципте на т.нар. архитектура, ориентирана към услуги, като няма наличен общоприет и добре дефиниран подход за подобна адаптация на системи, ориентирани към услуги. В статията е описано проучване на механизмите за самоадаптация на софтуерни системи, като е анализирана приложимостта им системи, ориентирани към услуги и са посочени съответните абстрактни нива, където това може да се реализира на практика.