International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:8, 2014

Service-Based Application Adaptation Strategies: A Survey

Sahba Paktinat, Afshin Salajeghe, Mir Ali Seyyedi, Yousef Rastegari

***Abstract*—**Service Oriented Architecture (SOA) allows modeling of dynamic interaction between incongruous providers, which enables governing the development of complex applications. However, implementation of SOA comes with some challenges, including its adaptability and robustness. Dynamism is inherent to the nature of service based applications and of their running environment. These factors lead to necessity for dynamic adaptation. In this paper we try to describe basics and main structure of SOA adaptation process with a conceptual view to this issue. In this survey we will review the relevant adaptation approaches. This paper allows studying how different approaches deal with service oriented architecture adaptation life-cycle and provides basic guidelines for their analysis, evaluation and comparison.

***Keywords*—**Context-aware, Dynamic Adaptation, Quality of Services, Service Oriented Architecture, Service Based Application.

I. Introduction

ALONG with the appearance and rapid improvement of SOA (Service Oriented Architecture), it has been one of the most typical platforms and the most promising computing paradigm in the internet era [4]. Applications are more and more build as a composition of services running on large scale, dynamic and heterogeneous environments. Software systems dealing with distributed applications in changing environment normally require re-configuration and troubleshooting to continue operation in all conditions. Changes during operation may stem from software system's self like failures or context, therefore detecting significant changes, deciding how to react and finally acting to execute decisions, are essential to improve lifetime of systems. In this setting, adaptability becomes a key feature of service as it provides a way for an application to continuously change itself in order to satisfy new requirements [1]. By self, we mean the whole body of the software, mostly implemented in several layers, while the context encompasses everything in the operating environment that affects the system's properties and its behavior [3]. Dynamic loosely bound systems make the management of large-scale distributed applications increasingly complex. Adaptation are necessary to keep the system within well-defined boundaries such as desired behavior because there are critical dependencies between SBAs (Service Based Applications) themselves and the services they are exploiting, which could change without notice, therefore SBAs have to be able to adapt to these unforeseen changes. According to the [9] adaptive service based software systems have the capability of monitoring the changing system status, analyzing and controlling tradeoff among multiple QoS features, and adapting its service configuration to satisfy multiple QoS requirement simultaneously. In the other word adaptation can be defined as a process of modifying SBAs in order to satisfy new requirement and to fit new situation dictated by the environment on the basis of adaptation strategies designed by the system integrator [1]. To advance the systems adaptation and management capabilities, existence of methods and tools is necessary to assess and possibly certify adaptation properties, not only at design time but also at run time. In this paper we try to describe basics and main structure of adaptable service base software systems with a conceptual view to this issue. To understand better these phenomena, in this survey we will study and review the relevant adaptation approaches. The remainder of this paper is organized as follow. Section II presents a classification of the principles related to the problem of service adaptation. Section III presents a review of existing works in the area. Section IV presents analysis of selected adaptive systems based on the classification presented in Section II. Finally, Section V concludes the paper.

S. Paktinat is studying with the Computer Engineering Department, Islamic Azad University-South Tehran Branch, Tehran, Iran (e-mail: sahbap67@gmail.com).

A. Salajegheh is with the Department of Computer, Islamic Azad University-South Tehran Branch, Tehran, Iran (e-mail: salajeghe@iau.ac.ir, afshinsala@yahoo.com).

M. A. Seyyedi is with the Computer Engineering Department, Islamic Azad University-South Tehran Branch, Tehran, Iran (e-mail: ma_seyyedi@azad.ac.ir).

Y. Rastegari is with the Electrical Engineering and Computer Engineering Department, Shahid Beheshti University, Tehran, Iran (e-mail: rastegari.yousef@gmail.com, y_rastegari@sbu.ac.ir).

II. Adaptive Service-Based System Principles

In this section we propose a review of adaptation concepts. Adaptation process starts with some triggers which notice that some changes have been happened in the system itself or in the environment. In [1] these triggers are classified as: changes in the infrastructural layer of the application due to quality of service changes; changes of the application context and location; changes of the user types, preferences, and constraints that require application customization and personalization as a means to adapt the application behavior to the particular user; changes in the functionalities provided by the component services that require modifying the way in which services are composed and coordinated; and changes in the way the service is being used and managed by its consumer, which leads to changes in the application requirements. Changes may arise from software system' self (internal causes) or context (any information that can be used to characterize the situation of an entity).

When one or some of these triggers change system's conditions, adaptive system have to be able to indicate and analyze them. Analyze of changes determine the adaptation strategies which define the possible ways to achieve those requirements given the current situation. The adaptation process is organized according to the IBM's MAPE-K (Monitor, Analyze, Plan, Execute and Knowledge) reference model of an autonomic system [11]. *Monitoring* is the observation function to detect changes, *analyze* to find adaptation strategy as a solution for adaptation goals, *planning* to compute a schedule of actions to satisfy adaptation strategy, and finally *execution* to perform those actions. Fig. 1 illustrates the adaptation taxonomy, which classifies adaptation concepts by answering to the following questions: "Why?", "When?", "What?", "Where?", "How?", and "Who". "Why?" describes the motivation of adaptation, "What?" clarifies the subject of monitoring and respectively the way it is described, temporal aspect of changes are addressed by "When?" questions, "Where?" explains the location of problem that needs to be resolved by adaptation, "How?" represents the way adaptation approach is delivered, and finally "Who?" addresses the level of automation and human involvement in self-adaptive software.

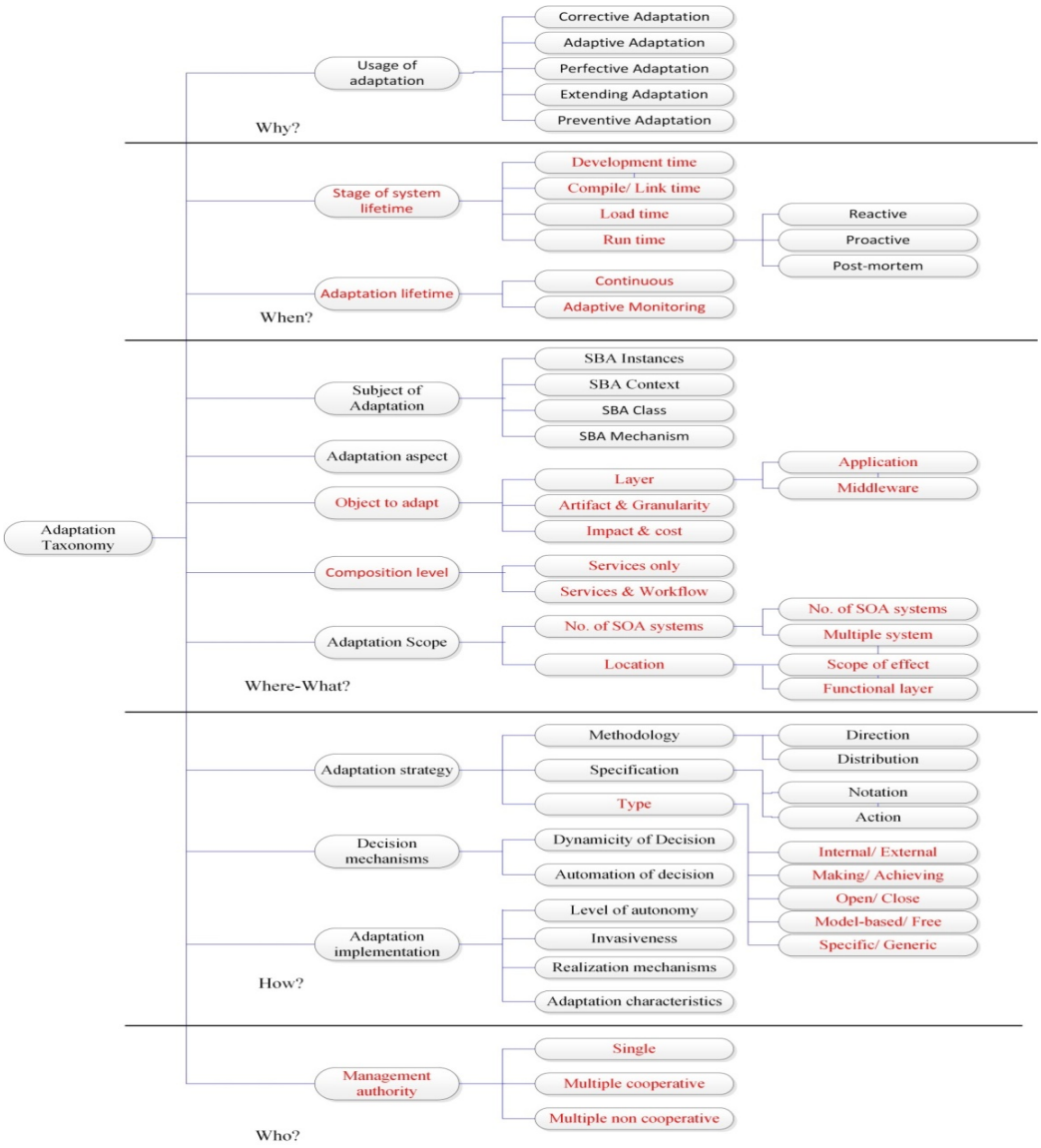

Fig. 1 Adaptation taxonomy

The taxonomy is an extension of the model proposed in S-Cube. The parts showed in red are perceived from concepts recommended in [3] and [11]. Adaptive properties are often known as self-* properties. In [3] these properties are categorized in three levels. First level which is mentioned as general level contains global properties of self-adaptive

software consists of *self-managing, self-governing, self-maintenance, self-control, self-evaluating,* and *self-organizing.* Second level is introduced as major level, which covers the answers of "why?" dimension in Fig. 1. The properties of this level are: *self-configuration, self-repairing, self-optimizing,* and *self-protection.* Finally the last level or primitive level, consist of *self-awareness,* which is based on self-monitoring, *self-situated,* and *context awareness*, which refers to operational environment. Due to the different principles of adaptation, comparison of related works becoming as more as complex and unsuitable, since we need methods and tools to assess certify adaptation properties. Some works have tried to address the evaluation of self-adaptive software. Villegas [2] proposes a framework for evaluating quality-driven self-adaptive software system. This framework is based on a set of adaptation properties derived from control theory properties. This framework defines a mapping between adaptation properties and software quality attributes. Then, it identifies a set of metrics used to evaluate software quality attributes. In our paper we chose this framework to evaluate selected adaptation approaches.

III. Related Works

Because of the dynamic and predictable nature of business application and distributed systems, delivering quality services to meet user demands is a big challenge. Quality management is a crucial element as it ensures that systems meet their requirements with respect to specific performance metrics. QoS is defined [4] as a measure of the fulfillment of the service agreed upon. QoS is a broad concept that encompasses multiple nonfunctional properties, or dimensions, some of which can be service specific and others more general such as availability, responsiveness and reliability.

Many researchers have defined many kinds of QoS attributes and models in the past few years. Choi's work represents the relationship of unique features of services and some quality attributes [10]. This paper first analyzes the service quality requirements and defines quality attributes of the quality model. Then defines metrics for each quality attributes. As already stated QoS management, is a main trigger for adaptation. Reconfiguration is one the most common strategies to address QoS management. For example, Li et al [4], presents an approach for QoS driven dynamic reconfiguration of the SOA based software. This approach can reconfigure SOA-based software to comply with new QoS constraints by replacing its individual or multiple component services. This approach, calculates the QoS attributes based on four service composition structures, who are considered as: Sequence structure, Parallel structure, Selection structure, and Loop structure. According to these composition structures, when the new QoS constraints are given, this approach tries to replace one or multiple services to comply with it. Replacement is done based on services critical factor. Critical factor is introduced to show the contribution of the QoS of each component service from the same software to the QoS of the software. This approach is based on modifying the current configuration parameters of the system. According to the Fig. 1, this approach is an adaptive adaptation that its necessity is to accommodate to the changes in the context.

Services are inherently dynamic and cannot be assumed to be always stable, thus in the case of service composition the failure of a single service leads to error propagation in the other service involved, and therefore the failure of the system. In these cases we need to remove the faulty behavior of a system by replacing it by a new version that provides the same functionality. This kind of adaptation is named as corrective adaptation in Fig. 1 and most of the time is a response to the *self-healing* property. Some of research activities have been done in this area. Lin [5] proposes an approach for replacing faulty services and some of their neighboring services to maintain the original end-to-end QoS constraint. It uses an iterative algorithm to search for a reconfiguration region that has replaceable services to meet the original QoS constraint for the region. In this approach, reconfiguration region may be replaced using one-to-one, one-to-many, or many-to-one service mapping. In this model, every service is classified by their functionalities into service classes. Every service in a service class has the same functionality, input and output types, but may deliver a different QoS. The reconfiguration strategy is based on the three process structure, contains; parallel, conditional and loop, which are same as [4]. The approach has implemented the adaptation manager in the, an SOA middleware, the intelligent Accountability Middleware Architecture (Llama). On the other word the study reported in this paper is on the part of service process reconfiguration after faulty services are identified.

Another popular adaptation strategy is re-compose, which means modifying the way services are composed. A service composition combines several services together to achieve a certain goal [1]. Most of the time the related works of re-composition are along with service selection/recommendation context. For example in [7] Madkour et al. proposes a three-phase adaptation approach. The method firstly selects the suitable services to the current context and then recommends them to the adaptation process, in the adaptation phase performs adaptation by using fuzzy sets represented with linguistic variables and memberships degrees to define the user's context and the rules for adopting the policies of implementing a service. Finally it deals with the complex requirements of the user by the composition of the obtained adaptable atomic services. In the approach, adapting is selecting the best policy for service implementing and recursively new service (re-compose services) to new utilization context. The adaptation method in [8] describes fuzzy based service adaptation middleware (FSAM) that can be used in context-aware middleware. The method formulates the service adaptation process by using fuzzy linguistic variables and memberships degree to define the context situation, similar as previous approach. The method, proposes three fitness functions to calculate the fitness degree for each policy and the current context situation. The decision for service adaptation is achieved by selecting the policy with the largest fitness degree.

IV. Analysis of Adaptation Approaches

To leverage the capability of self-adaptive systems, it is necessary to validate adaptation mechanisms to ensure that self-adaptive software systems function properly and users can trust them [2]. After reviewing adaptation principles and introducing some related works, in this section we will compare and analyze these works according to the model which proposed in [2]. The model defines a mapping between adaptation properties and software quality attributes, and then it identifies a set of metrics used to evaluate software quality attributes. The analyze process results of this model are summarized in Table I, and we present the characterization of our selected adaptive approaches based on this table.

TABLE I
Applying the Characterization Model to Selected Adaptive Approaches

Approach	Adaptation Goal	Reference Outputs	Measured Outputs	Control Actions	Adaptation Properties	Evaluation	Metrics
Lin et al.[5]	Self- healing/ self-recovery	Contracts: QoS Constraints defining computational states	SLOs	Discrete operations affecting the managed system's software architecture	Short settling time, termination, stability, accuracy	Simulated environment to evaluate performance	Reconfiguration time, Re-composition time
Ying Li et al. [4]	QoS preservation self-configuration	Contracts: QoS	SLOs, Logical properties of computational elements	Discrete operations affecting the managed system's software architecture	Stability, termination , Short settling time	None	None
Madkour et al. [7]	Self-configuration / re composition	Contracts: QoS	Logical properties of computational elements	Discrete operations affecting the managed system's software architecture	Stability, termination	None	None
Cao et al. [8]	Self-configuration / re-composition	Single reference values	Logical properties of computational elements	Discrete operations affecting the managed system's software architecture	Stability, termination, accuracy	Running examples to evaluate effectiveness	None

The paper, suggests a model to characterize self-adaptive software, which consisting of eight analysis dimensions. These dimensions include; 1) adaptation goal, the main reason for the approach to be self-adaptive, 2) reference inputs, specific set of values specified the state to be achieved by the adaptation mechanism. Reference inputs are specified as: single reference values; some form of contracts (e.g., QoS, service level agreements (SLA), or service level objectives (SLO)); constraints defining computational states; or even functional requirements, 3) measured outputs, 4) computed control actions, are characterized in MAPE-K loop and in practical by the nature of the output of adaptation planner, 5) system structure, 6) observable adaptation properties, 7) proposed evaluation, and finally 8) identified metrics and key performance indicators. The model categorizes adaptation properties as follows: stability, accuracy, short settling time (i.e., the time required for the adaptive system to achieve the desired state), small overshoot, robustness, termination, consistency, scalability, and security.

As already stated, adaptation in SBA may be motivated by variety of factors, or triggers. Such triggers may concern the component services or the context of SBA. Each trigger can be associated with a set of adaptation strategies that are suitable to re-align the application within the system and/or context requirements [6]. Relationships between adaptation triggers and adaptation strategies are presented in [6]. The work identifies changes in the service functionality, service quality, business context, computational context, and finally user context. Table II, lists our selected adaptation approach's triggers and strategies.

TABLE II
Relationship between Adaptation Triggers and Adaptation Strategies

Approach	Adaptation triggers	Adaptation strategy
Lin et al.[5]	Changes in the service functionality/failure	Reconfiguration/ substitution
Ying Li et al. [4]	Changes in the service quality	Reconfiguration/ substitution
Madkour et al. [7]	Changes in the user context	Re-composition/ re-selection
Cao et al. [8]	Changes in the computational context (network rate, network delay)	Reconfiguration/ substitution

V. Conclusion

Service Oriented architecture is still a new and active research area, and self-adaptive software enjoys a growing importance. In this paper we declared different aspects of adaptation taxonomy which were not covered in S-Cube. The key to understanding adaptation taxonomy is recognizing the relationships between the adaptation properties and software quality attributes that were illustrated in Table I.

The most important aspect of adaptation is adaptation strategy. Each adaptation strategy can be characterized by its complexity and its functional and non-functional properties. The identification of the most suitable strategy is supported by a reasoned that also bases its decisions on multiple criteria extracted from the current situation and from the knowledge obtained from previous adaptations and executions, so specifying the relationships between adaptation triggers and strategies and the current situation like context is too important to improve adaptation process. This work explains some of existing approaches that contain different strategies based on special triggers.

REFERENCES

[1] Raman Kazhamiakin, Salima Benbernou, Luciano Baresi, Pierluig iPlebani, Maike Uhlig4, and Olivier Barais. (2010). Adaptation of Service-Based Systems. In *S-cube.* springer.

[2] Norha M. Villegas, Gabriel Tamura and Rubby Casalla. (2011). A framework for evaluating quality-driven self-adaptive systems. New York: 6th Internation Symposium on Software Engineering for Adaptive and Self-Managing Systems.

[3] M. Salehi and L.Tahvildari. (2009). Self-adaptive software: Landscape and research challehges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*.

[4] Ying Li, Xiaorong Zhang, YuYu Yin, and Jian Wu. (2010). Qos-driven dynamic reconfiguration of the SOA based software. *The IEEE International Conference on Service Science(ICSS)*, (pp. 99-104). Hangzhou

[5] Kwei-Jay Lin, Jing Zhang, Yanlong Zhai and Bin Xu. (2010). The design ad implementation of service process reconfiguration with end-to-end QoS constraints in SOA. *Service Oriented Computing and Application*, 157-168.

[6] Antonio Bucchiarone, Cinzia Cappiello, Elisabetta Di Nitto, Raman Kazhamiakin, Valentina Mazza, Marco Pistore. (2009). Design for adaptation of service-based applications: Main issues and requirements. In *Service_oriented Computing. ISCOC* (pp. 467-476). Stockholm: Springer Berlin Heidelberg.

[7] Mohcine Madkour, Driss El Ghanami , Abdelilah Maach and Abderrahim Hasbi. (2013). Context-aware service adaptation: An approache based on fuzzy sets and service composition. *Information Science and Engineering*, 1-16.

[8] Jiannong Cao, Na Xing, Alvin T.S Chan, Yulin Feng, Beihong Jin. . (2005). Service adaptation using fuzzy theory in context-aware mobile computing middleware. *11th IEEE International Embedd and Real-Time Computing Systems and Applications*, (pp. 496-501).

[9] S.S Yau, N.Ye, H. Sarjoughian and Huang. (2008). developing service-based software system with QoS monitoring and adaptation. *IEEE International Workshop on Future Trends of Distributed ,* (pp. 74-80). omputing Systems.

[10] Si Won Choi, Jin Sun Her and Soo Dong Kim. (2007). Modeling QoS attributes and metrics for evaluating services in SOA considering consumer's perspective as the first class requirement. *2nd IEEE Conference on Asia-Pacific Service Computing*, (pp. 398-405). Tsukuba.

[11] Valeria Crdellini, Emiliano Casalicchio, Vincenzo Grassi, Stefano Lannucci, Francesco Lo Presti, Raffaela Mirandola. (2012). Moses: A framework for QoS driven runtime adaptation of service-oriented systems . *IEEE Transaction on Software Engineering*, (pp. 1138-1159).