State of Runtime Adaptation in Service-oriented Systems: What, Where, When, How and Right

Leah Mutanu [1], Gerald Kotonya [2*]

[1] School of Science and Technology, United States International University, P.O. Box 14634-00800, Nairobi, Kenya
[2] School of Computing and Communication, Lancaster University, Lancaster LA1 4WA, United Kingdom
[*]lmutanu@usiu.ac.ke

Abstract: Software as a Service reflects a "service-oriented" approach to software development that is based on the notion of composing applications by discovering and invoking network-available services to accomplish some task. However, as more business organizations adopt service-oriented solutions and the demands on them grow, the problem of ensuring that the software systems can adapt fast and effectively to changing business needs, changes in their runtime environment and failures in provided services has become an increasingly important research problem. Dynamic adaptation has been proposed as a way to address the problem. However, for adaptation to be effective several other factors need to be considered. This paper identifies the key factors that influence runtime adaptation in service-oriented systems, and examines how well they are addressed in 29 adaptation approaches intended to support service-oriented systems.

1. Introduction

Service-Oriented Architecture (SOA) provides the conceptual framework for realizing service-oriented systems (SOS's) by supporting dynamic composition and reconfiguration of software systems from networked software services [1]. Rosen [2] identifies the key motivations for SOA as agility, flexibility, reuse, integration and reduced cost. However, the need to ensure that the systems can adapt quickly and effectively to changing business needs, changes in system quality and changes in their runtime environment is an increasingly important research problem [3]. Effective adaptation ensures the system remains relevant in a changing environment and is an accurate reflection of user expectations.

Taylor [4] defines dynamic adaptation as the ability of a software system's functionality to be changed at runtime without requiring a system reload or restart. Taylor points out that there is an increasing demand for non-stop systems, as well as a desire to avoid annoying users. However, current approaches for supporting runtime adaptation in service-oriented systems differ widely with respect to the nature of systems they support, the types of system changes they support and their underlying model of adaptation [5][6]. In addition, it is also unclear how these approaches address the important issue of ensuring the adaptation is effective. A growing consensus amongst researchers is that runtime adaptation in SOA should incorporate a validation element [16][18].

In their research roadmap for self-adaptive systems, Lemos et al. [7] emphasize the need for feedback control in the life cycle of self-adaptive systems, and the need to perform traditional design-time verification and validation at runtime. In another survey, Salehie et al. [45] note that testing and assurance are probably the least focused phases in the engineering of self-adaptive software. Papazoglou et al. [18] echo this view. They note that the bulk of research in adaptive service-oriented systems has focused largely on dynamic compositions. Adaptation validation goes beyond verifying that the adaptation conforms to its operational specification. Validation is concerned with verifying the acceptability of an adaptation, often from the point of view of the system user – i.e. is it the right adaptation for the problem as opposed to whether it is specified right? Validation assesses the effectiveness of an adaptation. Because user requirements are constantly changing, a self-validation process would enable the adaptation system to self-assess and self- evolve in order to remain relevant.

This paper identifies the key research challenges and the factors that influence runtime adaptation in service-oriented systems. The influencing factors are used as basis for reviewing 29 approaches intended to support runtime adaptation in service-oriented systems. The survey compliments existing surveys and extends our earlier survey [72] to include an in-depth review of runtime validation in service-oriented systems.

Notable additions in this survey include a detailed review of the different models and techniques used to support runtime validation in service-oriented systems, when they are applied, their primary focus and the different strategies employed. Eighteen service-oriented approaches that support runtime validation are reviewed in this context. This survey has three objectives (i) to provide an overview of the key challenges in runtime adaptation for service-oriented systems; (ii) to propose a simple, but effective scheme for assessing runtime adaptation approaches in service-oriented systems; (ii) to provide an overview of the state of runtime adaptation approaches in service-oriented systems.

The paper is organised as follows, Section 2 outlines the key research challenges for runtime adaptation in service-oriented systems. Section 3 identifies the key factors that influence runtime adaptation in service-oriented systems and review how well they are supported in 29 service-oriented approaches. Section 4 provides some concluding thoughts and a look ahead.

2. Research Challenges

A number of research initiatives are investigating effective ways to improve on runtime adaptation in service-oriented systems. These initiatives are however inadequate for addressing the issues identified in Section 1 for the following reasons:

- *Static adaptation rules.* Current approaches for supporting runtime adaptation in service-oriented systems are based on rules that reconfigure systems based on fixed decision points which do not take into account the dynamic nature of the factors that influence adaptation[45] [7]. Indeed, Di Nitto et al. [76] attribute the dynamic nature of software to the fact that requirements cannot be fully gathered upfront and cannot be "frozen". Thus while various studies have been conducted to address the challenge of adapting software to address the ever changing requirements, currently, no single solutions to this problem exists. Existing research revolves around the "local" adaptation of specific cases. Di Nitto et al. highlight the need for research to devise technologies and methods to enable crosscutting adaptations.

- *Poor support for validation*. Current approaches for supporting runtime adaptation in service-oriented systems offer poor support for validation [16] [7]. Like most autonomic systems, runtime adaptation in service-oriented systems is based on IBM's *Monitoring, Analysis, Planning, and Execution* model (MAPE)[47]. However, MAPE does not support validation. The lack of mechanisms for validating adaptation make it difficult to gauge the appropriateness and effectiveness of adaptation decisions, and limit our understanding of the nature of problems for which they are suited. Validation provides an avenue for adaptation rules to evolve and remain relevant because the factors that influence adaptation are constantly changing[72]. Validating adaptation goes beyond verifying that the adaptation conforms to its operational specification. Validation is concerned with verifying the acceptability of an adaptation [77], often from the point of view of the system user, i.e. "is it the right adaptation for the problem?" as opposed to "is it specified right?" Validation assesses the effectiveness of an adaptation.

- *Poor support for diversity*. Current approaches for runtime adaptation are built around predefined changes requests and adaptations, and are often embedded within the applications they support. This limits their extensibility, portability and the quality of adaptation they offer. For example Cubo et al. [6] and Tanaka and Ishida [32] describe approaches that are concerned with specific application contexts. Swaminathan [5] and Cardellini [16] propose models that promote context variability, however the author provides no information about the implementation or evaluation of the models. There is no evidence that the approaches support diversity.

- *Poor support for proactive adaptation*. Most approaches to adaptation in service-oriented systems are reactive [50] [74]. They recompose the system as a reaction to change rather than anticipate change. While reactive adaptation has the advantage of requiring only a small set of recent system conditions to select an adaptation, allowing for a timely decision, it has a number of limitations. First, reactive adaptation is based largely on static system properties and conditions that do not take into account previous aspects of system behaviour that may inform better adaptation selection. Secondly, reactive adaption lags behind current system conditions, which may be short-lived or change as the adaptation is being carried out resulting in unnecessary adaptations that may impact system quality. Lastly, the inability to anticipate change makes it difficult to address disruptive system changes such as service and quality failures in a timely manner.

These key challenges represent the gaps in the runtime adaptation of service-oriented systems and are highlighted in this paper through a review of existing work in this area. This paper aims at highlighting the importance of addressing these gaps when developing dynamic self-adaptive service oriented applications.

3. Factors that influence adaptation

Most of the work on self-adapting software systems takes inspiration from control theory and machine learning. Control-theory splits the world into a controller and a plant. The controller is responsible for sending signals to the plant, according to a control law, so that the output of the plant follows a reference (the expected ideal output). Figure 1 shows a typical control loop. Although it is difficult to anticipate when and how change occurs in software systems, it is possible to control when and how the adaptation should react to change.

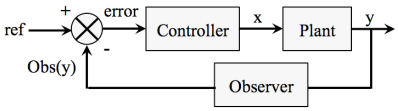

Fig. 1. *Dynamic physical system*

Dynamic adaptive systems require information about the running application as well as control protocols to be able to reconfigure a system. For example, keeping web services up and running for a long time requires collecting of information about the current state of the system, analyzing that information to diagnose performance problems or to detect failures, deciding how to resolve the problem (e.g., via dynamic load-balancing or healing), and acting on those decisions. Figure 2(a) shows the control process for a software system equivalent of the physical system shown in Figure 1. The *controller* maps onto an *adaptation* process that reconfigures the runtime system to address the changing needs in its application context. Figure 2(b) show how the adaptation process can be improved using validation. Validation tracks, assesses and adjusts adaptations to ensure that they reflect user expectations.

Lemos et al. [7] highlight the importance of understanding the factors that influence adaptation. They posit that this helps in the comprehension of how software processes change when developing self-adaptive systems. They describe these factors as design decisions pertinent to self-adaptive systems. These are: observation, representation, control, identification, and adaptation mechanism. These decisions however do not include validation, which they state is key the key to ensuring that the software system satisfies functional requirements and

meets their expected quality attributes. The key challenges with current approaches include defining models that can represent a wide range of system properties and the need for feedback control loops in the life cycle of self-adaptive systems and self-validation. The nature and quality of runtime adaptation in service-oriented systems is influenced by system changes (i.e. adaptation triggers), the nature of the application and the logical area where it executes (i.e. application context), the strategy used to reconfigure the system in a particular change context (i.e. adaptation model), and the effectiveness of the adaptation (i.e. validation). Together, these factors, represent the *what*, *where, when* and *how*, and *right* of runtime adaptation. It is also important to note that these factors constantly shift and evolve making it difficult to specify adequate adaptation rules in advance.

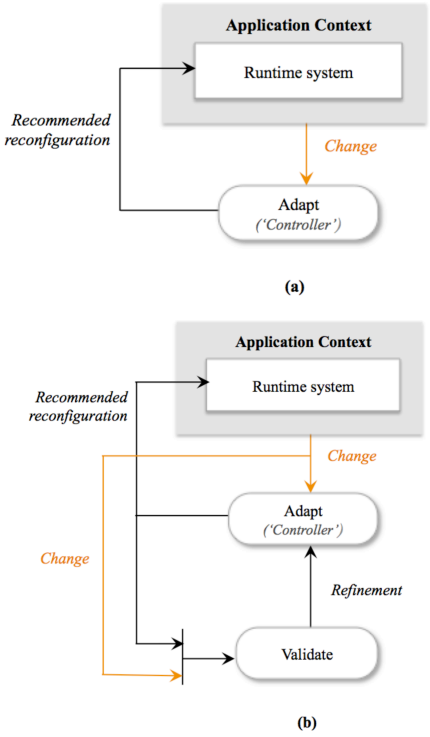

Fig. 2. *Dynamic software system*

Hirschfeld et al. [38] suggest that runtime adaptation in service-oriented systems should address the *what*, *when*, and *how* of adaptation (Figure 3). In Hirschfeld et al. [38], the *what* distinguishes between the basic properties of the system's computation, state, and communication. The *when* addresses the time when adaptations can be made operational in the system during software development i.e. at development time, compile-time, load-time, or runtime. The *how* studies tools and techniques that allow for adaptations to become effective. The *what*, *when* and *how* defined by Hirschfield et al. relate adaptation to system properties, software development stages, and tools to effect adaptation. This is different from our classification, which relates *what*, *where*, *when* and *how,* and *right* to change triggers, application context, adaptation model and validation [72]. Our classification is supported by a recent review of the state of runtime adaptation in service-oriented systems reveal that there are other important factors (dimensions) such as the application context, adaptation triggers, and validation.

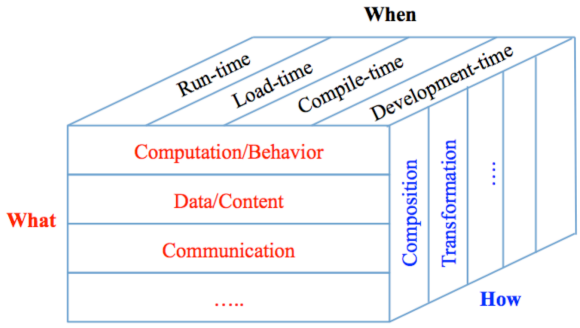

Fig. 3. *Adaptation dimensions*

Paktinat et al. [75] present a similar taxonomy of adaptation strategies for service-based systems. They classify adaptation according to *what* should be monitored, *when* the change should occur, *where* the problem is located, *how* adaptation is delivered, *why* adaptation should occur and *who* should be involved. Paktinat et al. do not distinguish triggers, provide no distinction between the different implementation models and do not support validation. Like [38] they also do not examine the effectiveness of the adaptation process. A comparison between the terminology and scope of our approach, and Hirschfeld et al. [38] and Paktinal et al. [75] is shown in Table 1. The elements of our approach are discussed in detail next.

Table 1. Runtime Adaptation Taxonomies

Reference	Trigger (cause)	Logical execution area	When carried out	Strategy	Validation	Focus
[38]	-	What	When	How	-	Development process
[75]	What	Where	When	How, Why, Who	-	System
This work	What	Where	When	How	Right	System

3.1. Change Trigger (What)

A change trigger represents *what* causes adaptation and the reason for it. Change triggers are a function of changes in the business environment, service failure, and changes in the system quality and its runtime environment.

- *Business Environment Triggers*. Changes in the business environment that the system supports may trigger adaptation. This may be caused by changes in user requirements, business rules or platform. Zeng [8] describes an adaptation approach that accepts changes in user requirements and business rules on the fly and composes services to address them. Similarly Cubo [6] describes an approach that uses changes in the system context and platform to trigger adaptation. Because user requirements are not static and constantly change at runtime, any adaptation solution should monitor the business environment and adapt the system accordingly.

- *Service Provision Trigger*. Failures in provided services, for example, incompabilities that impact on service composition, network outages and poor service quality,

could trigger adaptation. The quality of a service-oriented system depends not only on the quality of the provided service, but on the interdependencies between services and resource constraints imposed by the runtime environment. This type of corrective adaptation is typical of self-healing systems. Robinson et al., [10] describes an approach that uses a consumer-centred, pluggable brokerage model to track and renegotiate service faults and changes. The framework provides a service monitoring system, which actively monitors the quality of negotiated services for emergent changes, SLA violations and failure. A similar approach, *The Personal Mobility Manager*, described in [29] emphasizes the need for automatic system diagnosis to detect runtime errors. It helps car drivers find the best route in or between towns, by suggesting optimal combinations of transportation according to local situations, such as traffic level, weather conditions and opening hours. These examples demonstrate that in addition to changes in functional requirements, service-oriented systems should also be monitored for failures in service quality to ensure that they meet the expected service level agreements.

- *Runtime Environment Triggers*. Changes in the runtime system can also trigger adaptation. Interacting services may impose dependability as well as structural constraints on each other (e.g. performance, availability, cost, and interface requirements). Dustdar et al. [9] describe a self-adaptation technique for managing the runtime integration of diverse requirements arising from interacting services, such as time, performance and cost. Swaminathan et al. [5] propose an adaptation approach based on self-healing as a means for addressing runtime system errors. Runtime resource contentions between services in the orchestration platform can result in significant falls in service quality. This emergent quality of service is difficult to anticipate before system composition, as resource demands are often dynamic and influenced by many factors. Newman and Kotonya [11] proposes a resource-aware framework that combines resource monitoring with dynamic service orchestration to provide a runtime architecture that mediates resource contentions in embedded service-oriented systems. Embedded systems typically operate in resource-constrained environments and often find application in isolated locations. Small resource changes in their operating environment can have significant impact on the system quality hence making them difficult to manage. It is therefore important to monitor the runtime environment and dynamically adapt to ensure embedded systems run smoothly.

Effective adaptation must address the real cause rather than the symptom. Taiani [13] describes this as a key challenge in adaptive fault tolerant computing. Moyano et al. [14] describe a system that monitors service failure and runtime environment triggers. These are changes in hardware and firmware, including the unpredictable arrival or disappearance of devices and software component. For example, a low memory trigger may be the result of an SLA violation or runtime environment resource failure. The resolution to the problem might involve replacing the service with a more efficient alternative or optimizing the runtime environment, or both. It is important that the adaptation process is not only able to find a good fit for the problem, but the right fit.

It is worth noting that adaptation triggers are not mutually exclusive; there is often significant overlap between them. For example, when a user is trying to access a travel assistant the user's environment acts as the source of the trigger. Additionally if the user is using a mobile device, which has limited memory resources, then the runtime environment acts as the source of another trigger. A service provision trigger will arise from the quality of service required by the user. Triggers can also invoke other triggers and therefore overlap. For example a user accessing the application from an urban, industrial, or affluent geographic location will often require services of high reputation and may not have devices with resource constrains. The users environment (geographic location) then invokes the runtime and service provision triggers. A change or failure may be the symptom of an unseen change or failure. Effective adaptation must therefore address the real cause rather than the symptom.

3.2. Application context (Where)

An application context defines nature of the application and the logical area where it executes. It helps us understand *where* adaptation takes place and the constraints involved. Cubo et al. [6] discuss the importance of creating adaptive systems sensitive to their application context (i.e. domain, location, time and activity). Tanaka and Ishida [32] identify an input language and a target language as the application context for a language translation application. They however do not provide evidence that their approach can be used in a different application context. Most of the approaches surveyed in this paper were concerned with specific application contexts. Zeng et al. [8], for example, describe a runtime approach for supporting business change in the automotive industry. Similarly Newman et. al. [11] describes an adaptation framework specifically for embedded resource-constrained environments. Baresi et al. [15] describes an adaptation framework specifically for a smart home system. Specific application contexts contain few data which is easy to process for decision making. Generic application contexts however contain a lot of information beyond what is actually needed and this is difficult to process. Most of the work that addresses specific application contexts does not provide insight into how such systems would work in a different application or where different triggers existed.

In their description for the *DigiHome* architecture Romero et al. [41] discuss the integration of multi-scale entities where different application contexts are addressed. In the *DigiHome* scenario, they consider several heterogeneous devices that generate isolated events, which can be used to obtain valuable information and to make decisions accordingly. They make use of Complex Event Processing (CEP), to find relationships between a series of simple and independent events from different sources, using previously defined rules. CEP is useful in getting better information at real time in generic applications. A few other approaches, including Swaminathan et al. [5], Cardellini et al. [16], and Zeng et al. [17] propose generic application contexts, but they only provide sketchy implementation details. Some approaches promote context variability. For example, Swaminathan et al. describe a context-independent, self-configuring, self-healing model for web services.

However, the author provides no information about the implementation or evaluation of the model. Huang et al. [20] describe an approach for developing self-configuring services using service-specific knowledge. They evaluate their approach on three different systems (i.e. a video streaming service, an interactive search service, and a video-conference service). However, it is evident from their discussion that the context needs to be known before the application is deployed.

The survey distinguished between dynamic adaptation approaches that are intended to support specific application contexts and generic solutions that can tailored for different application contexts. Examples of approaches intended for specific application contexts include Zeng et al [8] and Autili et al [28] whose work target automotive and manufacturing domains. Examples of generic approaches included Swaminathan [5] and Cardellini et al [16]. Most approaches surveyed are designed for specific application contexts.

3.3. Adaptation model (When and How)

An adaptation model indicates *when* the adaptation process is carried out and *how* the model is implemented in relation to the system it manages. A decision on when to conduct adaptation is arrived at depending on when the adaptation requirements are known as well as the availability of the requirements for adaptation.

This survey focuses runtime adaptation. This corresponds to situations where the requirements are only known after the system has started executing. This is the typical situation in ubiquitous and mobile computing scenarios. The availability of the requirements for adaptation, such as system resources can also determine when to conduct adaptation. For example, if the resources are available online then dynamic adaptation can be conducted; otherwise it can be pushed to a later time when they will be available. Table 1 provides a summary of current approaches for runtime adaptation. Papazoglou et al. [18] and Baresi et al. [3] identify the key techniques that can be used to achieve runtime adaptation as self-configuring, self-healing, and self-optimizing techniques.

- *Self-Configuring* is the automatic re-composition of services to adapt to changes in the service environment. The work of [19], [8] and [21] describe self-configuring adaptation techniques.
- *Self-Optimizing* is the automatic re-composition of services to improve quality of a service. The work of [9], [17], and [13] describes self-optimizing adaptation techniques.
- *Self-Healing* is the automatic re-composition of services to address a service failure. Self-healing techniques detect system malfunctions and initiate policy based corrective actions without disrupting the runtime environment [18].

Romay's [21] review of self-adaptation techniques in SOA reveals that current research focuses largely on self-configuring techniques. There is very little research on self-optimizing or self-healing techniques. Bucchiarone et al. [22] note that focusing on only one technique limits the effectiveness of the approach. Our survey focuses on two aspects of the *Adaptation Model* - the nature of its implementation (i.e. pluggable vs. embedded) and the strategy adopted to effect the implemented technique (i.e. reactive vs. predictive). An implementation may be associated with any of the many adaptation techniques. We believe that this high-level view provides a more transferrable and reusable description of the underlying adaptation model.

3.3.1. Adaptation strategy – Predictive vs. reactive: Adaptation can occur in response to anticipated changes (predictive) or in response to change trigger (reactive). Reactive adaptation controls and adapts the environment according to the users' situation. The system perceives its environment through sensors and reacts to changes as they occur. An ideal predictive approach does not take into account the reactions of the system under control, but only the environment under which it operates. The control of the environment is not pegged on observing the reactions of the user. As a result such an approach should offer quick response and high performance with few sensors required.

Tanaka and Ishida [32] propose a model that focuses on predicting the executability of services (i.e. if a message request will cause execution failure). Unfortunately they provide limited detail on the implementation and evaluation of their approach. In their event-driven quality of service prediction approach, Zeng et al. [8] point out that most adaptation approaches focus on monitoring Quality of Service (QoS) constraints and as such cannot provide early warning to prevent QoS degradation. They describe a model that makes use of data mining and prediction scoring to anticipate change. However, they provide only limited information on its evaluation. Wang et al. [24] proposes a predictability model based on the Q-Learning algorithm using the Markov Decision Processes. They explain that human oriented services are rarely predictable. They point out that many service properties keep changing in a manner that prior knowledge of these changes may not be available. Instead they suggest incorporating reinforced learning in adaptation techniques to ensure that adaptation techniques remain relevant. Their model uses a decision process that maximizes the expected sum of rewards. While predictive adaptation shows some promise, there is very little research on them and even less information on their evaluation. Most of the research is reactive with adaptation taking place after triggers have occurred, making it difficult to mitigate unforeseen catastrophic results.

3.3.2. Model implementation: An adaptation model can be implemented as an intrinsic part of the system it manages or as a pluggable framework that monitors change variables and effects re-composition from outside the system. Garlan et al. [23] depict a pluggable approach where the adaptation module is plugged on to legacy systems. In their work an external model is used to monitor and modify a system dynamically.

Most of the initiatives surveyed however adopted an embedded approach. Zeng et al. [8] and Cubo et al. [6] are typical of this approach. However, there is growing acknowledgement amongst researchers that a pluggable approach offers a better engineering solution. Pathan et al. [63] propose a generic approach to context aware modeling through the use a separate component for context reasoning. Garlan et al. [23] state that, the use of external control mechanisms for self-adaptivity is a more effective engineering solution than localizing the solution. A pluggable engine can be analyzed, modified, extended, and

reused across different systems. Most solutions presented in existing literature were embedded which limits their reusability and portability.

3.4. Summary

Table 2 shows our results of surveying 29 approaches that provide runtime adaptation for SOS's. It is important to mention that the survey was intended to be representative rather than exhaustive. The approaches were carefully selected to provide a good coverage of current adaptation in service-oriented systems. To ensure representative coverage all selected approached support runtime adaptation and provide some level of support for at least two factors that influence runtime adaptation.

Each approach is reviewed in terms of the nature and extent of support for change triggers, adaptation model, validation and application context. Most of the approaches provide limited support for runtime and service quality triggers. However, they provide comparatively good support for business environment triggers. Only Ivanovic et al. [12] describes an approach for supporting all the three adaptation triggers. In their work they talk of the computational cost of service networks as being dependent on internal and external factors. They recognize that triggers for adaptation are due to overlapping factors that are both internal and external to the service. Of the approaches reviewed, only a few provide support for adaptation validation. However, the support is very limited. There is poor support for diversity with most approaches designed to support specific application contexts. This limitation may be related to the fact that most of the approaches are embedded. Of the approaches surveyed only Zeng et al. [17] provides a detailed discussion of the adaptation techniques used to address quality of service issues that arise from interacting services (i.e. system concerns).

Table 2. Summary of adaptation approaches

	Adaptation Trigger			**Adaptation Model**		
				Strategy	**Implementation**	
Approach	Runtime environment	Service provision	Business environment	Reactive (R) Predictive (P)	Embedded (E) Pluggable (P)	**Application Context** (G= Generic S= Specific)
Zeng et al [8]	○	○	◒	R	E	S
Swaminathan [5]	○	◒	◒	R	N/A	G
Cubo et al [6]	○	○	●	R	E	S
Huang et al [20]	○	○	◒	R	E	S
Dustdar et al [9]	◒	○	◒	R	E	S
Autili et al [28]	◒	○	◒	R	E	S
Cardellini et al [16]	◒	◒	◒	R	P	G
Lorenzoli et al [29]	○	◒	○	R	P	S
He et al [30]	○	◒	○	R	E	S
Mateescu et al [31]	○	○	◒	R	P	S
Zeng et al [17]	●	◒	●	P	E	G
Robinson et al [10]	○	●	○	R	P	S
Tanaka et al [32]	○	○	●	P	E	S
Siljee et al [33]	○	●	●	R	E	S
Orriens et al [34]	○	○	●	R	E	S
Wang et al [24]	○	○	●	R	E	S
Sliwa et al [35]	○	○	●	R	E	S
Lin et al [36]	○	○	●	R	E	S
Ivanovic et al [12]	●	●	●	P	E	S
Hussein et al [37]	○	◒	○	R	P	S
Hirschfield et al [38]	○	●	○	R	E	S
Tosic et al [39]	○	◒	○	R	P	S
Maurer et al [40]	○	●	○	R	P	S
Romero et al [41]	○	○	●	R	P	S
Li et al [42]	◒	◒	○	R	P	S
Newman et al [11]	●	○	○	R	P	S
Motahari-Nezhad et al [43]	○	○	●	R	E	S
Cugola et al [44]	○	●	○	R	E	S
Andre et al. [46]	◒	◒	◒	R	P	G

Key

● - Supported
◒ - Weakly Supported
○ - Not Supported N/A - Not Applicable

Most of the approaches surveyed provide strong support for dynamic adaptation, which is not surprising as they are intended to support runtime change. However, most of them are implemented as part of the application they manage (i.e. embedded) rather than pluggable. Pluggable approaches include [16], [29], [31], [32] and [36]. The approach proposed by Swaminathan at al. [5] does not provide adequate implementation details, so it is unclear how it is implemented (i.e. as a pluggable or embedded solution).

3.5. Support for runtime validation (Right)

A typical adaptation process uses a predefined decision model to select an appropriate adaptation in response to a change trigger. This relationship is often predefined and stored as a set of adaptation rules. However, the dynamic nature of service-oriented systems means these factors are constantly changing, which makes it difficult to specify adequate adaptation rules *a priori*. This is further complicated by the likelihood of competing adaptation requests. This means that rules used to inform adaptation decisions cannot be static and must constantly evolve to remain relevant. Most approaches that support runtime adaptation are based on rules that reconfigure systems based on fixed decision points. This means that most adaptations in service-oriented systems are responses to change rather than anticipation. One way to address the problem is through the validation of adaptation decisions. As mentioned earlier validation refers to building the right product based on the user's product acceptance [77]. The most challenging concern for validation is uncertainty. This highlights the need to conduct it at runtime unlike conventional approaches that focus on conducting it before deployment. Because self-adaptation targets environments with hard to predict, highly dynamic, and comparatively resource-constrained conditions, a more inclusive validation approach would address this challenge. Such an approach should perform validation for the entire control loop against some optimization objective. This would include validating the reconfiguration process, the executability of the reconfiguration process, and the new configuration, against the business, application and runtime environment.

Validation serves two key roles. First, it provides a mechanism for assessing the effectiveness of an adaptation decision i.e. how well a recommended adaptation addresses the concerns for which the system is reconfigured. Secondly it provides us with insights into the nature of problems for which different adaptations are suited. Most autonomic systems are underpinned by IBM's *Monitoring, Analysis, Planning, and Execution* model (MAPE) [47]. Figure 4 illustrates a typical MAPE-K cycle.

Fig. 4. *MAPE-K Architectural Model for Autonomic Computing*

MAPE model for autonomic computing intelligent control loop works as follows:

- The *monitor function* provides the mechanisms that collect, aggregate, filter and report details on adaptation triggers. To detect the triggers sensors are used. For example a sensor could be used to detect the client device used and provide a report. Baresi and Guinea [27] describe service-monitoring approach as the support for the dynamic selection and execution of monitoring rules at runtime.
- The *analyze function* provides the mechanisms that correlate and model complex situations. To do this it analyses the report provided by the monitor and issues an alert if certain thresh hold values are reached based on preexisting rules. For most approaches surveyed, this is a simple process as it addresses only one type of change. The process however would become complex where multiple changes have to be monitored. Psaier et al [36] attribute the difficulty in management of adaptation of service-oriented applications to the changing interaction and behavior patterns that possibly contradict and result in faults from varying conditions and misbehavior in the network.
- The *plan function* provides the mechanisms that construct the actions needed to achieve goals and objectives. Most approaches surveyed focus on the identification of a suitable alternative service as can be seen in the work of Cervantes et al [34]. He et al [14] however propose the adaptation of web service composition based on workflow patterns re-composition.
- The *execute function* provides the mechanisms that control the execution of a plan with considerations for dynamic updates. It invokes the adaptation technique. For example it would call for the re-orchestration of the service-oriented application to make use of the suggested workflow pattern.

While the MAPE model is evident in many self-adaptive frameworks for SOA, it lacks a runtime mechanism for supporting validation. This is also evident in the survey we conducted as most researchers do not consider validation as part of the adaptation process. Kephart et al. [81] highlight the importance of continuously validating an autonomic system to gauge its effectiveness. A separate survey of current research in validating service-oriented systems reveals that various verification and validation techniques are used to support adaptation. These include formal methods, model-based, and machine-learning techniques. The next section discusses these techniques and some of the challenges faced.

3.5.1. *Formal methods:* Salehie et al. [45] suggest that formal methods can be used for verification and validation of self-adaptive software to ensure its correct functionality, and to understand its behavior. Weyns et al.[49] state that formal methods set out to show that a system has some desired properties by proving that a model of that system satisfies those properties. The use of formal methods can also be seen in the work of Fiadeiro et al. [82] who set out to develop models through which designers can validate properties of composite services. Assembly and binding techniques such as the ones provided by Service Component Architecture can then be used to put together heterogeneous service components. They define a mathematical model of computation and an associated logic for service-oriented

systems which preserves correctness. In their work semantics of service modules are defined after which they formulate a property of correctness that guarantees services programmed and assembled (as specified in a module) provide the business functionality advertised by that module. However, the model does not take into account shifting user requirements, the changeability of services and unpredictable runtime environments that are continuously evolving.

Armando et al. [83] propose a platform for the *Automated Validation of Trust and Security of Service-Oriented Architectures*. They emphasize that deploying services in network infrastructures such as SOAs entails a wide range of trust and security issues. Modeling and reasoning about these issues is complex because SOAs use different technology, can interfere with each other and run on unpredictable environments. They propose the use of a *validator* that takes any model of a system and its security goals and automatically checks whether the system meets its goals under the assumption that the network is controlled by a *Dolev-Yao* intruder (a formal model used to prove properties of interactive cryptographic protocols). As proof of concept they formalize ten application scenarios of SOAs from the e-Business, e-Government and e-Health application areas. While this work provides some good insight into the modeling of dynamic aspects of service-oriented systems, it is very closely concerned with validating aspects of security and trust. It is not easy to port the model to address other quality and system aspects in SOAs.

Another example of how formal methods are used can be seen in the work of Arcaini et al. [47] who model and validate a distributed self-adaptive service-oriented application. They design a traffic monitoring system with a number of intelligent cameras along a road and apply a formal modeling approach for self-validation. In their work a frame work for formal modeling, validating, and verifying distributed self-adaptive systems based on the multi-agent Abstract State Machines (ASM) formalism is presented. They state that formal methods can be used as a rigorous means for specifying and reasoning about self-adaptive systems' behavior, both at design time and at runtime. They however note that over-specification is a challenge with the formal approach due to the rigidity of the formalisms Timed Automata. This challenge can be avoided through separation of concerns where by one adaptation concern is handled at a time. Lemos et al. [7] also agree that this approach can be challenging to use at runtime. They state that formal methods can be too expensive to be executed regularly at run-time when the system adapts, due to their time and space complexity.

3.5.2. Model-based approaches: In this approach models check the behavior of a self-adaptive system during design and are later used to test the implementation during and after development. Gomaa et al. [50] use patterns to model how the components that make up an architecture pattern cooperate to change the software configuration at run-time. They propose a model-based run-time adaptation pattern for distributed hierarchical service coordination in service-oriented systems, in which multiple service coordinators are organized in a distributed hierarchical configuration.

Based on interpretations of UML Models as graphs and graph transformation systems, Baresi et al.[51], posit that the consistency between platform and application can be validated using model based approaches. In order to reason about planned or unanticipated reconfigurations of architectures, they use graph transformation rules to capture the dynamic aspects of architectural styles. As a case study they make use of the reference architecture for a supply chain management system that involves a consumer component, a retailer service, a warehouse service, a shipping service, and a manufacturer service. Their model of the architectural style supports the architect when deciding whether the style is suitable for his application.

However many researchers agree that the use of model based techniques on their own is inadequate. Fleurey et al. [54] combine model driven and aspect oriented techniques when validating dynamic adaptation. Aspect orientation provides modularization mechanisms to separate the crosscutting concern at the programming level. Model Driven Engineering (MDE) techniques on the other hand consider models as the primary development artefact and use them as a basis for obtaining an executable system in different ways. Fleurey combines both techniques by designing a base model and different variant architecture models at design time that are processed to produce a correct system configuration at runtime. The actual configurations of the application are built at runtime by selecting and composing the appropriate variants. Their work however looks at validating adaptation rules at design-time. Calinescu et al. [52] advocate for the use of both modeling techniques and mathematically based techniques to plan the adaptation steps necessary to identify requirement violations at runtime. They however point out model learning as a key challenge of their work. In their discussion on the use of models at runtime for self-assurance, Cheng et al. [53] highlight some key challenges with the approach. A key issue in this approach is to keep the run-time models synchronized with the changing system. They recommend the use of probability distribution functions, the attribute value ranges, or using the analysis of historical attribute values. More advanced and predictive models of adaptation are needed for systems that could fail to satisfy their requirements due to side effects of change.

3.5.3. Machine learning: In order to assess the effectiveness of an adaptation decision a self-adapting system needs to learn. The learning process can yield results that can be used to update the adaptation process with a goal of remaining relevant. Alpaydin [27] defines Machine learning as programming computers to optimize a performance criterion using example data or past experience. He further explains that machine learning is used where human expertise does not exist and the solution changes with time. Learning occurs by building models that are good and useful approximations from examples of data provided. To achieve this statistics are used to make inferences from the examples of data provided and efficient algorithms are used to solve optimization problems as well as represent and evaluate generalized models.

Machine learning algorithms can generally be categorised as supervised, unsupervised and reinforced learning. Supervised learning algorithms make predictions

based on a set of examples. Classifiers, Decision trees, Neural Networks, and regression are some examples of supervised learning that can be seen in the work of Hoffert et al [58]. Unsupervised learning occurs where labelled examples are not available. The goal is to organize the data in some way or to describe its structure. This can mean grouping it into clusters using algorithms such as k-mean clustering or Expectation-maximization (EM) clustering. For the unsupervised approach EM clustering is often considered because it provides better optimization than distance-based or hard membership algorithms, such as *k-Means*. EM easily accommodates categorical and continuous data fields making it the most effective technique available for proper probabilistic clustering. Skałkowski et al. [25], recommend the dynamic adaptation of services using machine learning. They show how a clustering algorithm can be used to provide automatic recognition of similar system states and grouping them into subsets (called clusters), based on information provided by the Monitoring element interface. Reinforced learning provides a method for the machine to quantify its performance in the form of a reward signal. Markov decision Processes are used to model the problem to be solved as seen in the work of Jureta et al. [59] and Wang et al. [24].

Experiments with EM clustering however show some gaps that are typical of natural data. In order to reinforce unsupervised learning, a supervised learning approach can be used such as neural network classification. Roohi [55] point out that most of the problems that prop up in all the fields of human operations pertain to organizing objects or data in different categories or classes. The challenge therefore is to assign an object or data item to a class based on a number of observed attributes (features) related to that object. Reinforced learning on the other hand is based on exploration that relies on a trial and error process. This presents safety challenges in risky application contexts as well as learning challenges where rewards are sparse. According to Cybenko [56] and Hornik et al. [57] artificial neural networks are good classification options as they have been able to approximate any function with good accuracy. Artificial neural networks, being nonlinear models, can be used to model any real world complex process.

Hoffert et al. [58] supports the idea of supervised machine learning in dynamic environments in maintaining of Quality of Service. They list possible supervised machine learning techniques to include decision trees, neural networks, and linear logistic regression classifiers that can be trained on existing data to interpolate and extrapolate for new data. Jureta et al. [59], in a similar study state that parameters such as quality of service, deadline, reputation, cost, and user preferences can be used as criteria in learning algorithms. Bayesian probabilities have also been used to express evidence about stakeholders' satisfaction in terms of degrees of belief. Schumann and Gupta [60] proposed a validation method to calculate safety regions for adaptive systems around the current state of operation based on a Bayesian statistical approach.

Many experiments have shown that deep neural networks are particularly good with natural data such as speech, vision, or language, which exhibit highly nonlinear properties, [61]. Najafabadi et al. [62] state that stacking up non-linear feature extractors (as in Deep Learning) often yield better machine learning results, e.g., improved classification modelling, better quality of generated samples by generative probabilistic models, and the invariant property of data representations. However they also point out that a downside of adaptive deep belief network is the requirement for constant memory consumption. Additionally the slow learning process associated with a deep layered hierarchy of learning data abstractions and representations from a lower-level layer to a higher-level layer makes it challenging to use at runtime.

In addition to the foregoing validation techniques, the nature of involvement, control mechanism and strategy adopted are also important. The next section reviews these three factors.

3.5.4. *Involvement – online vs. offline validation:* Validation can be performed at design-time, runtime or during system maintenance. Traditionally this has been conducted offline at design-time. However at this stage validation can only address requirements that were known during development. The shift towards self-adaptive systems called for validation to be performed dynamically at runtime – i.e. online validation. This however introduces the challenge of ensuring that that the recommended adaptation is timely, right and has adequate system resources available to support it. This section examines some representative involvements.

There have been attempts to use self-test mechanisms at runtime to validate the changes. King et al. [64], recommend that dynamically adaptive behavior in autonomic software should include rigorous off-line and on-line testing. They propose an in-built test manager in autonomic systems to support this. Zhang et al. [65] propose a run-time model checking approach for the verification of adaptation. Salehie et al. [45] identifies a key challenge posed by the online approach as the presence of several alternatives for adaptable artifacts and parameters in the system. They note that this leads to several paths of execution in different scenarios. Further the dynamic decision-making approach makes it even more complex. Cardellini et al. [16] present an architecture that calls for the validation of the adaptation decision off-line. This can be achieved by collecting statistics based on past adaptation decisions.

Autili et al. [28] propose a model-based solution for self-adapting context-aware services. They provide methodologies to generate adaptable code from UML service models during development. Model-To-Code transformations are performed by means of a code generator offline. They perform both *online validations* (to generate test cases, before the service execution, by taking into account both the service model and the service code) and *off-line validation* (whilst the service is running and uses the generated test cases). They however only give an overall description but no real world case studies that would validate the whole framework.

The sole use of design time (model based approaches) for self-validation is not adequate. Dustdar et al. [9] recommend combining both design time and runtime management to build evolvable systems. In their work, model-driven development techniques are first adopted and adapted to support the modeling and design of compliant Web services and processes at design time. They conclude that *Online* and/or *offline* monitoring components must be

introduced as well as tools such as a dashboard to allow the human users to observe the system and react on problems and critical situations. However, the goal of autonomic computing is to eliminate human intervention. A self-learning approach could provide one solution.

3.5.5. Control mechanism: Control mechanisms have been at the heart of engineering practice for several decades now. The purpose of a controller is to produce a signal that is suitable as input to the controlled plant or process, [66]. A key requirement for any self-adaptive system is to make use of control values that tell the system how to adapt. A self-validating system signals the need to take corrective action whenever the output of the adapter deviates from expectations. This deviation is referred to as the tracking error.

Tamura et al. [67] describe feedback control loops as validation that depends on online measurements on past performance from the target system and the adaptation mechanism. They posit that measured outputs are important for making adaptive system quality decisions at runtime. Dustdar et al. [9] present a solution that incorporates a model-driven compliance support, runtime interaction mining, run-time management of requirements, and explicit control-loop architecture for self-validation. They develop a Web service information model to provide a holistic view of past and present requirements associated with services. Then, based on these requirements explicit *feedback-control* techniques are used to perform adaptation strategies.

Feed-forward control techniques on the other hand take environmental or external context into account i.e. the current situation. This can also provide validation information of the adaptation process. Cardozo et al. [68] propose a feed forward approach to validate the dynamic adaptation of software. Their approach uses a symbolic execution engine to reason about the reachable states of the system, whenever contexts are activated or deactivated. Context activation and deactivation requests are allowed depending on the presence of erroneous states within reachable states. Fredericks et al. [69] point out that traditional testing techniques treat inputs and expected outputs as fixed, static values throughout the testing process. However, requirements specification, and the environment can change and thereby cause input and expected output values to no longer be representative test cases. This would make a feed forward approach inadequate.

3.5.6. Strategy: The strategies used for validation in self-adaptive service-oriented systems are either reactive or proactive as shown on Table 3, with reactive strategies being most common. Hielscher et al. [70] describe reactive approaches as those that trigger validation based on monitored events. Consequently validation occurs after monitoring. In a proactive approach validation occurs before monitoring, and is based a predictive model trained on historical data. The objective of a proactive approach to validation is to avoid the cost of an unsatisfactory adaptation process. Achieving this however is not a simple task as it calls for dealing with uncertainty. A more common approach to validation is the reactive approach, which is easier to implement, but may need the adaptation process to iterate severally before an acceptable decision is arrived at. The reactive and proactive adaptation processes discussed in section 3.3.1 are independent of the validation strategies discussed in this section. For example a reactive adaptation approach that occurs before triggers are fired can be validated proactively or reactively. It is validated proactively if past validation behaviour is used to determine the acceptable action before a trigger is detected. The validation decision then waits for the trigger. On the other hand it could be validated reactively after if validation occurs after the trigger is detected. This process waits for the adaptation decision to be arrived at, after which user feedback could be used to validate adaptation. Most of the work reviewed provided implementation details on either adaptation (Table 2) or validation (Table 3) making it difficult to compare the relationship between both strategies.

Fleurey et al. [54] propose a reactive model-based approach to self-validation, which includes invariant properties, and constraints that allow the validation of the adaptation rules at design time. During runtime, the adaptation model is processed to produce a correct system configuration that can be executed. This is achieved through monitoring of the system state and the execution context (such as memory, or CPU usage, or available network bandwidth, or battery level) after which adaptation rules are triggered. Validation then occurs by comparing the woven model with the reference model. It is worth noting that validation is reactive because it occurs after monitoring. Similarly Baresi et al. [51] also proposal a reactive approach to validating the adaptation of service oriented systems. However they recommend runtime validation rather than design time validation. They state that in the dynamic world of service-oriented architectures, what is guaranteed at development time may not be true at run time. They advocate for a reactive approach by arguing that it is virtually impossible to predict all the evolutions and changes that might occur in the services we use, and the same is true for the environment. Therefore, monitoring allows them to take notice of infringements of expectation and react to them.

Hielscher et al. [70] outline some of these consequences as loss of money, unsatisfied users, and reduced system performance. Autili et al. [28] propose a proactive strategy that explores how to validate extra-functional issues during the service development and execution. They state that Bayesian Reliability Models and Queuing Networks can be analyzed at development time to validate the Service Model characteristics and a decision made available at run-time on how adaptation of the service will occur for the detected execution context. Validation at design time is therefore performed to generate test cases, before the service execution, by taking into account contextual information and possible changes of the user needs. When a service is invoked, a run-time analysis is performed (on the available models) and, based on the validation results; a new set of models is selected. Validation then occurs before adaptation is triggered based on past performance.

Another proactive approach is presented by Hoffert et al. [58] who point out the difficulty in maintaining the Quality of Service (QoS) properties (such as reliability and latency) in dynamic environments such as disaster relief operations or power grids. They state that the challenge arises from the slow human response times, and the complexity of managing multiple interrelated QoS settings.

Table 3. Support for self-validation in service-oriented systems

Approach	Technique/mechanism	Involvement (when applied)		Primary focus		Control mechanism		Strategy
		Off line	Online	Verification	Validation	Feedback	Feed forward	
King et al. [64]	Regression testing	●	●	○	●	●	○	Reactive
Fleurey et al. [54]	Model Based Techniques	●	○	●	○	○	●	Reactive
Baresi et al. [51]	Model based	●	○	●	○	○	●	Reactive
Autili et al. [28]	Model Based - Bayesian Reliability Models	◒	◒	○	◒	◒	◒	Reactive
Arcaini et al. [47]	Formal Modeling	●	○	●	●	●	○	Proactive
Dustdar et al. [9]	Model-driven approach	◒	◒	○	◒	◒	○	Reactive
Morin et al. (2009)	Model Based Techniques	●	○	○	●	○	●	Reactive
Salehie et al. [45]	Formal methods	○	◒	◒	◒	○	◒	Reactive
Hoffert et al. [58]	Decision tree, Artificial Neural Net-work, and Linear Logistic Regression Classifier	○	●	●	○	●	○	Proactive
Skałkowski et al. [25]	Clustering algorithm	○	●	○	●	○	●	Reactive
Jureta et al. [59]	Markov Decision Processes	○	●	●	○	●	○	Reactive
Cardellini et al. [16]	Model-Based, Linear programming	●	●	●	●	●	○	Reactive
Gomaa et al. [50]	Model-based	○	●	●	○	○	●	Reactive
Cardozo et al. [68]	Static symbolic exploration	○	●	●	○	○	●	Proactive
Wang et al. [24]	State monitoring & Formal Specification	○	◒	◒	○	○	◒	Reactive
Bartolini et al. [73]	Statistical metric	○	●	○	●	○	●	Reactive
Hielscher et al. [70]	Regression Testing	◒	○	○	○	◒	○	Proactive
Weyns et al. [74]	Model based	○	●	●	○	○	●	Reactive

Huber et al. [71] also emphasize the need for a proactive approach to self-validation. They state that in order to provide QoS guarantees (e.g, availability and performance) for virtualization and Cloud Computing environments; the ability to predict at run-time how the performance of running applications would be affected is required. They refer to it as online performance prediction, which allows for the proactive adaptation of the system to the new workload conditions, thereby avoiding SLA violations or inefficient resource usage. Their work describes how they use software performance models to predict the effect of changes and to decide which actions to take.

From our survey of self-validating techniques conducted it is evident that very few researchers use machine-learning techniques to evaluate the effectiveness of an adaptation solution. Further, the work on machine learning techniques does not evaluate several algorithms to improve on accuracy. Additionally, most of the research work on dynamic adaptation does not integrate both runtime and static validation. There is also very little evidence to show that the validation techniques can work for different adaption triggers or different adaptation techniques.

Additionally the survey of self-validation techniques for adaptation in service-oriented systems presented in Table 3 shows that majority of the work on validation does not provide adequate details on how adaptation occurs. As a result most of the work shown in Table 3 is not presented in Table 2, which reviews work on self-adaptation. This observation reinforces the widely held view that validation is poorly supported in service-oriented adaptation [18], [16], [45], [7]. King et al. [64] for example, provide details of a self-validating approach for testing adaptive Autonomic computing systems. They however do not describe how the adaptive system works and simply refer to the MAPE architectural blue print for Autonomic Computing put forward by IBM. This makes it challenging to adequately review the adaptation aspect of their work in order to understand how adaptation works. As a result they do not appear in Table 2, which focuses on dynamic adaptation. However they provide adequate details for a Self-Testing Framework. This is expected because self-testing is the focus of their work and as a result we review them in Table 2.

Only a handful of approaches tackle the issue of validation in self-adaptation. Dustdar et al. [9] describe a self-adaptation technique for managing the runtime integration of diverse requirements arising from interacting services, such as time, performance and cost. They also recommend combining both design time and runtime management to build evolvable systems. They note that current work in adaptive systems provides no integrated support for validating design rules, which affect both the design time and runtime of a system. Although they describe both adaptation and validation they do not provide adequate implementation details on how they work. Cardellini et al. [16] propose an approach to adaptation and validation, however only sketchy details are provided on adaptation. They emphasize the importance of service failure and changes in system quality as triggers for adaptation but only

provide a general plan of how their proposed adaptation should work. On the other hand they provide detailed descriptions of an architecture that calls for the validation of the adaptation decision off-line. This can be achieved by collecting statistics based on past adaptation decisions.

4. Conclusions

The paper has discussed the importance of runtime adaptation in SOS's and identified the design decisions that must be made when developing these systems. These decisions describe the *what, where, when and how* and *right* of adaptation. Specifically adaptation triggers tell us what cause adaptation, the application context tells us where to adapt, the adaptation models tell us when and how to adapt and validation tells us how effective the adaptation is.

We have used these factors to review the current state of runtime adaptation in service-oriented systems. Our survey reveals that most of the approaches provide patchy support for the key factors that influence adaptation. Most adaptation approaches are tied to particular application contexts, focus on specific aspects changes and are embedded in the application they manage. It is also clear that there is limited empirical evidence to indicate the effectiveness of the approaches reviewed. Lastly, we have provided a possible solution that integrates and extends the strengths of current approach to support validation. We believe this paper makes a significant contribution towards understanding and addressing a challenging problem.

5. References

[1] Erl, T.: 'SOA Principles Of Service Design', Prentice Hall, 2008

[2] Rosen, M., Lublinsky, B. K., Smith, T., and Balcer, M. J.: 'Applied SOA: service-oriented architecture and design strategies', Wiley Publishing, Inc., 2008

[3] Baresi, L.: 'Self-adaptive Systems, Services, and Product Lines', SPLC. In Proc. of the 18th International Software Product Line Conference, Florence, Italy, September 15-19, 2014, pp. 2-4

[4] Taylor, R. N., Medvidovic, N., and Oreizy, P.: 'Architectural Styles For Runtime Software Adaptation', European Conference on Software Architecture, Joint Working IEEE/IFIP Conference European Conference on Software Architecture, , Cambridge, 2009, pp. 171 – 180

[5] Swaminathan, R. K.: 'Self-Configuring And Self-Healing Web Services In Complex Software Systems', CECS IT project report, Part of the Special Projects Group: University of Waterloo, Waterloo, 2008

[6] Cubo, J., Canal, C., and Pimentel, E.: 'Supporting context awareness in adaptive service composition',In Proc. of 1st Workshop on Autonomic and SELF-Adaptive Systems (WASELF'08), Gijón (Spain) 2008 (JISBD'08), Sistedes, 2008, pp. 64-73

[7] Lemos, R. de. et al.: 'Software Engineering for Self-Adaptive Systems: A Second Research Roadmap', Lecture Notes In Computer Science, Vol. 7475. Springer-Verlag, 2013, pp.1-26

[8] Zeng, L., Benatallah, B., Lei, H., Ngu, A., Flaxer, D., and Chang, H.: 'Flexible composition of enterprise web services', Electronic Markets, Volume 13, Number 2 (12)., 2003, pp. 141-152

[9] Dustdar, S., Goeschka, K., Truong, H., Zdun, U.: 'Self-Adaptation techniques for complex service-oriented systems', 5th Intl. Conf. Next Generation Web Services Practices, 2009, pp. 37-43

[10] Robinson, D., and Kotonya, G.: 'A runtime quality architecture for service-oriented systems', In Proc. of the 6th International Joint Conference on Service-Oriented Computing, Sydney, Australia, 2008, pp. 468-482

[11] Newman, P., and Kotonya, G.: 'Managing Resource Contention in Embedded Service-Oriented Systems with Dynamic Orchestration', In Proc of 10th International Conference on Service Oriented Computing, 2012, pp. 435-449.

[12] Ivanovic, D., Carro, M., and Hermenegildo, M.: 'Towards Data-Aware QoS-driven Adaptation for Service Orchestrations', IEEE International Conference on Web Services, 2010, pp. 107-114

[13] Taiani, F., and Fabre, J.: 'Some Challenges in Adaptive Fault Tolerant Computing', 12th European Workshop on Dependable Computing, Toulouse, France, 2009, pp. 3-pages

[14] Moyano, F., Baudry, B., Lopez, J.: 'Towards Trust-Aware and Self-Adaptive Systems', In Proc. of IFIPTM, 2013, pp. 255-262

[15] Baresi, L., Guinea, S., and Pasquale, L.: 'Service-Oriented Dynamic Software Product Lines', Computer, vol. 45, 2012, pp. 42-48

[16] Cardellini, V., Casalicchio, E., Grassi, V., Presti, F. L., Mirandola, R.: 'Towards self-adaptation for dependable service oriented systems', ICSOC LNCS vol. 5835, 2009, pp. 24-48

[17] Zeng, L., Lingenfelder, C., Lei, H., and Chang, H.: 'Event-Driven quality of service prediction', In Proc. of the 6th International Conference on Service-Oriented Computing, Sydney, 2008, pp. 147-161,

[18] Papazoglou, M. P., Traverso, P., Distar, S., and Leymann, F.: 'Service oriented computing: state of the art and research challenges', IEEE Computer Society. Vol. 40, 2007, pp. 38-45

[19] Madkour, M., El Ghanami, D., Maach, A., and Hasbi, A.: 'Context-aware service adaptation: An approache based on fuzzy sets and service composition', Information Science and Engineering, 2013, pp. 1-16

[20] Huang, A., and Steenkiste, P.: 'Building self-configuring services using service specific knowledge', In Proc of the IEEE International Symposium on High Performance Distributed Computing, IEEE Computer Society: Washington DC, USA, 2004, pp.45-54

[21] Romay, M., Fernández-Sanz, L., and Rodríguez, D.: 'A Systematic Review of Self-adaptation in Service-oriented Architectures', In Proc. The Sixth International Conference on Software Engineering Advances. Barcelona, Spain, 2011, pp. 331-337

[22] Bucchiarone, A., Marconi, A., Mezzina, C., Pistore, M., Raik, H.: 'On-the-Fly Adaptation of Dynamic Service-Based Systems: Incrementality, Reduction and Reuse', International Conference on Service Oriented Computing, Berlin, 2013, pp. 146-161

[23] Garlan, D., Schmerl, B., and Chang, J.: 'Using Gauges for Architecture-Based Monitoring and Adaptation', In Proc. Working Conference on Complex and Dynamic System Architecture. Brisbane, Australia, 2001

[24] Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W., Bouguettaya, A.: 'Adaptive service composition based on reinforcement learning'. In Proc. of 8th International Conference on Service-Oriented Computing, San Francisico, California, 2010, pp. 92-107

[25] Skałkowski, K., and Zieliski, K.: 'Automatic adaptation of soa systems supported by machine learning'. In Technological Innovation for the Internet of Things, *volume 394 of* IFIP Advances in Information and Communication Technology, pp. 61-68. Springer Berlin Heidelberg, 2013

[26] Schumann, J., Gupta, P., Jacklin, S.: 'Toward Verification And Validation Of Adaptive Aircraft Controllers' In: Proc. IEEE Aerospace Conference, IEEE Press, 200)

[27] Alpaydin E.: 'Introduction to Machine Learning'. The MIT Press, (2004).

[28] Autili, M., Berardinelli, L., Cortellessa, V., Marco, A., Ruscio, D., Inverardi, P., and Tivoli, M.: 'A development process for self-adapting service oriented applications', ICSOC, Springer, Heidelberg, 2007, pp. 442-448

[29] Lorenzoli, D., Mussino, M. P., Sichel A., and Tosi, D.: 'A SOA based self-adaptive personal mobility manager', In Proc. of IEEE International Conference on Services Computing, 2006, pp. 479 – 486

[30] He, Q., Yan, J., Jin, H., and Yang, Y.: 'Adaptation of web service composition based on workflow patterns', In Proc. of the 6th International Conference on Service-Oriented Computing, Sydney, Australia, 2008, pp. 22-37

[31] Mateescu, R., Poizat, P., and Salaün, G.: 'Adaptation of service protocols using process algebra and on-the-fly reduction techniques', In Proc. International Conference on Service-Oriented Computing, 2008, pp. 84-99,

[32] Tanaka, M., and Ishida, T.: 'Predicting and learning executability of composite web services', In Proc. of the 6th International Conference on Service-Oriented Computing, Sydney, Australia, 2008, pp. 572-578,

[33] Siljee, J., Bosloper, I., Nijhuis J., and Hammer, D.: 'DySOA: making service systems self-adaptive', In Proc. 3rd International Conference on Service-Oriented Computing, Amsterdam, The Netherlands, 2005, pp. 255-268

[34] Orriens, B., and Yang, J.: 'A rule driven Approach for Developing Adaptive Service Oriented Business Collaborations', In Proc. of the IEEE International Conference on Services Computing, Chicago, 2005, pp 182-189,

[35] Śliwa, J., Gleba, K., Amanowicz, M.: 'Adaptation Framework foR web services provision in tactical environment. Military Communications and Information Systems Conference', Wrocław, Poland, 2010, pp. 52-67

[36] Lin, K., Zhang, J., Zhai Y., and Xu. B.: 'The design and implementation of service process reconfiguration with end-to-end QoS constraints in SOA', Service Oriented Computing and Application, 2010 , pp. 157-168

[37] Hussein, M., and Gomaa, H.: 'An Architecture-Based Dynamic Adaptation Model and Framework for Adaptive Software Systems', 9th IEEE/ACS International Conference, 2011, pp. 165-172

[38] Hirschfeld, R., and Kawamura, K.: 'Dynamic Service Adaptation', In Proc. of Distributed Computing Systems Workshops', 2004, pp. 290-297

[39] Tosic , V., Ma, W., Pagurek, B., Esfandiari, B.: 'Web Service Offerings Infrastructure (WSOI) -A Management Infrastructure', In Proc. of NOMS (IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM), Seoul, South Korea, 2004, pp. 817-830

[40] Maurer, M., Brandic, I., Emeakaroha, V., Dustdar, S.: 'Towards knowledge management in self-adaptable clouds', Fourth International Workshop of Software Engineering for Adaptive Service-Oriented Systems (SEASS '10), Miami, Florida, USA, 2010, pp. 527-534

[41] Romero, D., Hermosillo, G., Taherkordi, A., Nzekwa, R., Rouvoy, R., and Eliassen, F.: 'The DigiHome Service-Oriented Platform', Software: Practice and Experience, Wiley-Blackwell, . (2013).pp. 1205-1218

[42] Li, G., Liao, L., Song, D., Wang, J., Sun, F., and Liang, G.: 'A Self-healing Framework for QoS-Aware Web Service Composition via Case-Based Reasoning', In Proc. of APWeb, 2013, pp. 654-661

[43] Motahari-Nezhad, H., Bartolini, C., Graupner, S., and Spence, S.: 'Adaptive case management in the social enterprise', In Service-Oriented Computing. Springer, 2012, pp. 550-557

[44] Cugola, G., Ghezzi, C., Sales Pinto, L., and Tamburrelli, G.: 'Adaptive Service-Oriented Mobile Applications: A Declarative Approach', In Proc. of the 10th international conference on Service-Oriented Computing, 2012, pp. 607-614

[45] Salehie, M., and Tahvildari, L.: 'Self-adaptive software: Landscape and research challenges', ACM Trans. Auton. Adapt. Syst. , New York, NY, USA, 2009, pp. 14

[46] Andre, F., Daubert, E., and Gauvrit, G.: 'Towards a Generic Context-aware Framework for Self-Adaptation of Service-Oriented Architectires', Intl Conf. on Internet and Web Applications abd Services (ICIW'2010), 2010, pp. 309-314,

[47] Arcaini, P., Riccobene, E., Scandurra, P.: 'Modeling And Analyzing MAPE-K Feedback Loops For Self-Adaptation', In Proc. of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '15, IEEE Press, 2015, pp. 13-23,

[48] Mutanu L. and G. Kotonya, G.: 'Consumer-centred Validation for Runtime Adaptation in Service-Oriented Systems', IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA'2016), 2016, pp.16-23,

[49] Weyns, D.: 'Towards an Integrated Approach for Validating Qualities of Self-Adaptive Systems', Workshop on Dynamic Analysis, 2012

[50] Gomaa H., and Hashimoto, K.: 'Model-Based Run-Time Software Adaptation For Distributed Hierarchical Service Coordination', In ADAPTIVE 2014, The Sixth International Conference on Adaptive and Self-Adaptive Systems and Applications pp. 1-6, 2014

[51] Baresi, L., Heckel, R., Thöne, S., Varró, D.: 'Modeling And Validation Of Service-Oriented Architectures: Application Vs. Style', In: Proc. ESEC/FSE and ACM SIGSOFT, ACM Press, 2003, pp. 68–77

[52] Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., and Tamburrelli. G.: 'Dynamic Qos Management And Optimization In Service-Based Systems', IEEE Transactions on SoftwareEngineering, 2011

[53] Cheng, B. H. C., De Lemos, R. et al.: 'Software Engineering for Self-Adaptive Systems: A Research Road Map', Springer-Verlag, 2009

[54] Fleurey, F., Dehlen, V., Bencomo, N., Morin, B., and Jézéquel. J.-M.: 'Modeling and Validating Dynamic Adaptation', In 3rd International Workshop on Models@Runtime (MODELS'08), France, 2008

[55] Roohi, F.: 'Artificial Neural Network Approach to Clustering', The International Journal Of Engineering And Science (Ijes), Volume 2, Issue 3, 33-38 Issn: 2319 – 1813 Isbn: 2319 – 1805, 2013

[56] Cybenko, G.: 'Approximation by superpositions of sigmoidal functions', Math. Control, Signals, Syst., vol. 2, no. 4, 1989, pp.303-314,

[57] Hornik, K., Stinchcombe, M., and White, H.: 'Multilayer feedforward networks are universal approximators', Neural Networks, 1989, pp.359-368,

[58] Hoffert, J., Mack, D., and Schmidt, D. C.: 'Using Machine Learning To Maintain Pub/Sub System Qos In Dynamic Environments', In Proc. of the 8th International Workshop on Adaptive and Reflective Middleware, 2009

[59] Jureta, I. J., Faulkner, S., and Thiran, P.: 'Dynamic Requirements Specification For Adaptable And Open Service Systems', Requirements Engineering Conference, 2007

[60] Schumann, J., Gupta, P., Jacklin, S.: 'Toward Verification And Validation Of Adaptive Aircraft Controllers', In: Proc. IEEE Aerospace Conference, IEEE Press, 2005

[61] Le, Q. V.: 'A Tutorial on Deep Learning: Nonlinear Classifiers and The Back propagation Algorithm', http://robotics.stanford.edu/~quocle/tutorial1:pdf, accessed on 11th August 2016, 2015

[62] Najafabadi, M. M., Villanustre, F, Khoshgoftaar, T. M, Seliya N, Wald, R, Muharemagic, E.: 'Deep learning applications and challenges in big data analytics', Journal of Big Data;2(1), 2015, pp. 1–21,

[63] Pathan, K. J., Reiff-Marganiec, S., Shaikh, A. A., and Channa, N.: 'Reaching Activities by Places in the Context-Aware Environments Using Software Sensors', Journal of Emerging Trends in Computing and Information Sciences, VOL. 2, NO. 12, ISSN, 2011, pp. 2079-8407

[64] King, T. M., Ramirez, A. E., Cruz, R. and Clarke. P. J.: 'An Integrated Self-Testing Framework For Autonomic Computing Systems', Journal of Computers, 2(9), 2007, pp. 37-249.

[65] Zhang, J., Goldsby, H., and Cheng. B. H. C.: 'Modular Verification Of Dynamically Adaptive Systems', In Proc. 8th International Conference on Aspect Oriented Software Development, 2009, pp. 161-172.

[66] Janert P. K.: 'Feedback Control for Computer Systems: Introducing Control Theory to Enterprise Programmers', O'Reilly Media, 2013

[67] Tamura, G., Villegas, N., M"uller, H. et al.: 'Towards Practical Runtime Verification and Validation of Self-Adaptive Software Systems', LNCS, vol. 7475, Springer, 2013, pp. 108-132

[68] Cardozo, N. Christophe, L. De Roover, C. and De Meuter, W.: 'Run-Time Validation Of Behavioral Adaptations', in International Workshop on Context-Oriented Programming , USA, 2014

[69] Fredericks, E. M., Ramirez, A. J., and Cheng, B. H. C.: 'Towards Run-Time Testing of Dynamic Adaptive Systems', In Software Engineering for Adaptive and Self-Managing Systems (SEAMS), ICSE, 2013, pp. 169–174.

[70] Hielscher, J. Kazhamiakin, R. Metzger A. and Pistore, M.: 'A Framework For Proactive Self-Adaptation Of Service-Based Applications Based On Online Testing', In Proc. First European Conf. Towards a Service-Based Internet, 2008

[71] Huber, N., Brosig, F., & Kounev, S.: 'Model-based self-adaptive resource allocation in virtualized environments', In Proc. of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, ACM, 2011, pp. 90-99

[72] Mutanu, L., Kotonya G.: 'What, Where, When, How and Right of Runtime Adaptation in Service-Oriented Systems', 2nd Workshop on Adaptive Service-Oriented and Cloud Applications, In Proc of 15th ICSOC Conference, Spain, Malaga, 2017

[73] Bartolini N, Bongiovanni G, Silvestri S.: 'Self-∗ through self-learning: Overload control for distributed web systems', Computer Networks, 2009, pp. 727-43.

[74] Weyns D, Iftikhar MU.: 'Model-based simulation at runtime for self-adaptive systems', In Autonomic Computing (ICAC), 2016 IEEE International Conference, 2016, pp. 364-373

[75] Paktinat S, Salajeghe A, Seyyedi MA, Rastegari Y. Service-based application adaptation strategies: a survey. International Journal of Computer, Electrical, Automation, Control and Information Engineering. 2014 Aug 1;8(8):1416-20.

[76] Di Nitto E, Ghezzi C, Metzger A, Papazoglou M, Pohl K. A journey to highly dynamic, self-adaptive service-based applications. Automated Software Engineering. 2008 Dec 1;15(3-4):313-41

[77] Sommerville, I.: 'Software Engineering', Pearson Education Limited, 2016, 10th Edition.

[78] Baresi L, Guinea S. Towards dynamic monitoring of WS-BPEL processes. In International Conference on Service-Oriented Computing 2005 Dec 12 (pp. 269-282). Springer, Berlin, Heidelberg.

[79] Psaier H, Dustdar S. A survey on self-healing systems: approaches and systems. Computing. 2011 Jan 1;91(1):43-73.

[80] Cervantes F, Ramos F, Gutiérrez LF, Occello M, Jamont JP. A new approach for the composition of adaptive pervasive systems. IEEE Systems Journal. 2018 Jun;12(2):1709-21.

[81] Kephart JO, Chess DM. The vision of autonomic computing. Computer. 2003 Jan;36(1):41-50.

[82] Fiadeiro J, Lopes A, Abreu J. A formal model for service-oriented interactions. Science of Computer Programming. 2012 May 1;77(5):577-608.

[83] Armando A, Arsac W, Avanesov T, Barletta M, Calvi A, Cappai A, Carbone R, Chevalier Y, Compagna L, Cuéllar J, Erzse G. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. InInternational Conference on Tools and Algorithms for the Construction and Analysis of Systems 2012 Mar 24 (pp. 267-282). Springer, Berlin, Heidelberg.