Chapter 10
Appraisal and Analysis of Various Self-Adaptive Web Service Composition Approaches

Doaa H. Elsayed, Eman S. Nasr, Alaa El Din M. El Ghazali and Mervat H. Gheith

Abstract Service-Oriented Requirements Engineering (SORE) plays a significant role in eliciting, specifying, and validating service requirements that will be developed by Web service technology. With the increasing complexity of users' requirements, Web services need to be combined together to fulfill them. The process of building new value-added services by integrating sets of existing Web services to satisfy users' requirements is called Web Service Composition (WSC). The main objective of WSC is to develop composite services to satisfy users' requirements, which does not only include Functional Requirements (FR), but also Non-Functional Requirements (NFR). One of the main challenges of WSC is how it deals with dynamic environments. Since the Web service properties and composition requirements are frequently changeable, this demands that SORE activities must be equipped with a self-adaptation mechanism to provide the most appropriate composite services and satisfy users' requirements emerged. Self-adaptation occurs in either a proactive or reactive manner. In this chapter, we appraise and analyze existing reactive adaptation research that deals with the problem of WSC in a dynamic environment in order to identify the research gaps in this field. These approaches are classified into three categories: used of variability models, context-awareness, and multi-agent approaches. Most of these approaches are not able to deal with continuous and unanticipated changes in complex uncertain contexts because they need to define the contexts in design time. It is usually

D.H. Elsayed (✉) · M.H. Gheith
Institute of Statistical Studies and Research, Cairo University, Cairo, Egypt
e-mail: doaa.hani@hotmail.com

M.H. Gheith
e-mail: mervat_gheith@yahoo.com

E.S. Nasr
Independent Researcher, Cairo, Egypt
e-mail: nasr.eman.s@gmail.com

A.E.D.M. El Ghazali
Sadat Academy for Management Sciences, Cairo, Egypt
e-mail: a.elghazali@gmail.com

M. Ramachandran and Z. Mahmood (eds.), *Requirements Engineering for Service and Cloud Computing*, DOI 10.1007/978-3-319-51310-2_10

difficult to predict all of the possible situations that might arise in an uncertain environment.

Keywords Web service composition · Reactive adaptation · User requirement

10.1 Introduction

Service-Oriented Architecture (SOA) is an architectural approach to design and develop distributed systems in the form of interoperable services. Interoperability is the ability of two or more systems to work together to achieve a common goal [1, 2]. A Web service is a technology that implements SOA [3]. Web services achieve interoperability between applications using three major Web technologies to provide an industrial standard for deploying, publishing, discovering, and invoking enterprises' services. The standard technologies for implementing Web services are Web Services Description Language (WSDL), Universal Description, Discovery and Integration (UDDI), and Simple Object Access Protocol (SOAP) [4]. With the increasing complexity of users' requirements, Web services need to be combined together to fulfill them [5]. The process of developing a composite service that satisfies users' requirements is called Web Service Composition (WSC). The ultimate objective of WSC is to develop composite services to satisfy users' requirements, and hence Requirements Engineering (RE) could be considered the most critical phase of WSC [6]. RE establishes the goals and objectives of the system in consultation with all relevant stakeholders. RE could be divided into Functional Requirements (FR) and Non-Functional Requirements (NFR) [7]. FR represent functionality in a system or component (i.e., what the system does). NFR are treated as requirements on quality of the system, such as Quality of Services (QoS), cost, scalability, usability, maintainability, etc. FR are represented by task/function, while NFR are operationalized by quality constraints. If FR and NFR are not defined correctly in the beginning, the resulting WSC will not fully satisfy a user's request.

RE has evolved from classical methods to object-oriented methods and finally to Service-Oriented Requirements Engineering (SORE) [8]. SORE defines methodologies to elicit, specify, and validate the services' requirements from two different standpoints: the service consumer and the service provider [8]. The service provider needs to understand the functional and non-functional parts of the service being offered. For the service consumers, the challenge is to find the best-matched service for the requirements while making a tradeoff among cost, FR, and NFR. One of the key research challenges of WSC is how WSC deals with dynamic environments. In dynamic composition environments, the change occurs during design and runtime, such as the availability of Web services, a composition of requirements, and changes in QoS (e.g., price, reputation, etc.) [9]. Therefore, WSC should be equipped with self-adaptation mechanisms to ensure the ability to adapt to meet changing requirements, and seek to minimize user interventions in order to provide

the most appropriate composite services and satisfy user's requirements [9]. SORE activities need to be performed at design time with more explicit constructs to specify requirements for Self-Adaptation Software (SAS), and are also needed for runtime adaptation for adaptable WSC approaches to deal with contextual changes in a dynamic environment [9]. SAS supports adaptation in either a proactive or reactive manner [10]. Proactive adaptation is able to predict the need for adaptation before the problem occurs [10]. Moustafa and Zhang [7] propose a proactive adaptation approach in WSC, which uses Markov Decision Process (MDP) to model WSC process and uses Q-learning for Reinforcement Learning (RL) technique to adapt to dynamic change in the WSC environments proactively. This approach monitors the WSC to determine proactive adaptation via analyzing the historical data in the Web service execution log. Aschoff and Zisman [11] also propose a ProAdapt framework for proactive adaptation in WSC. This framework triggers proactive adaptation in case of changes in response times of service operation or unavailability of operations in services and providers. It uses Exponentially Weighted Moving Average (EWMA) technique to predict response times of operations. The adaptation process occurs during the execution of WSC. Moustafa and Zhang [7] and Aschoff and Zisman [11] need to extend to support adaptive WSC in other types of QoS aspects and other circumstances, for example, the availability of new (better) service operations in comparison to the ones used in a composition, and changes in the structure of the WSC's workflow. Contrary to proactive adaptation, reactive adaptation is able to react to change; this means that adaptation occurs after an event which causes the need for adaptation [12].

This chapter presents various reactive adaptation WSC approaches to deal with the changes that might occur within and outside the dynamic composition environment; approaches are analyzed and compared. These approaches are classified into three categories: variability model, context-aware WSC, and multi-agent approaches. To the best of our knowledge, no survey on reactive adaptation WSC in the dynamic environment has been published yet. The rest of this chapter is organized as follows. Section 10.2 presents adaptation aspect and self-adaptation properties in WSC. Section 10.3 presents levels and challenges for RE for self-adaptive systems; Sect. 10.4 presents requirement specification models in WSC. Various self-adaptive WSC approaches are classified in Sect. 10.5; the comparison and limitation of approaches are presented in Sect. 10.6. Finally, Sect. 10.7 gives the conclusion and future work.

10.2 Self-Adaptive WSC

WSC needs to provide adaptive capabilities in order to respond to evolving demands and changes without compromising operational and financial efficiencies [13]. Avila [14] presents five of the main aspects that are considered parts of adaptation in WSC as shown in Fig. 10.1. The first aspect is the adaptation goal, which defines the adaptation purpose based on FR and/or QoS needs. Some

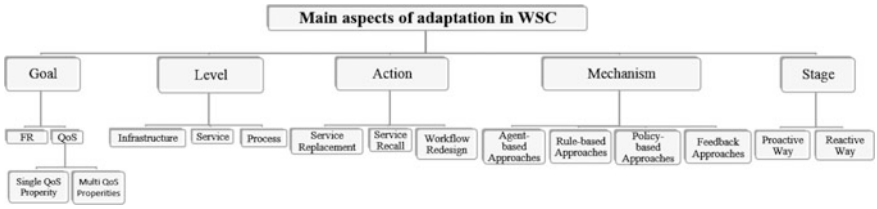

Fig. 10.1 The main aspects of adaptation in WSC

approaches such as Deng et al. [15] deal with single QoS optimization, while Liu et al. [16], Shanshan et al. [17], and Qiqing et al. [18] deal with multi QoS criteria. The second aspect is the adaptation level. These levels are identified differently in the literature. Raik [19] classifies them into three levels: infrastructure, service, and process. Various approaches concerned with process level are focused on in this chapter. The third aspect is adaptation action, which is used to solve an adaptation problem. This action can involve service replacement, workflow redesign, and service recall. The fourth aspect is the adaptation mechanism, which means the approaches that could be applied to execute an adaptation action such as agent-based, rule-based, policy-based, or feedback approaches. We focus on three adaptation mechanisms, namely variability model, context-aware, and multi-agent. The fifth aspect is the stage of adaptation, which means the time when the adaptation occurs. An adaptation could be triggered in a proactive or reactive way as explained before. Reactive approaches are focused on in this chapter.

Self-adaptation (self-*) properties are important in adaptive WSC too. Self-* properties enable WSC to deal with dynamic WSC execution environment. Figure 10.2 shows self-* properties applied to WSC. These self-* properties are self-healing, self-optimizing, self-configuring, and self-aware [14]. Self-healing is automatic discovery and correction of the failure of WSC by itself due to changes in QoS and/or FR without any human intervention and without stopping the WSC [20]. As WSC is done dynamically, they need to balance themselves with the changing environment. If the Web service cannot balance itself, then it leads to several faults such as incorrect order, misunderstood behavior, QoS service failure such as poor response and service unavailability, etc. [21]. Incorrect order occurs due to message flow through SOAP [22]. When the packets arrive in an order different on receiver side from sender side, this leads to incorrect order [22]. Misunderstood behavior occurs when the requester receives a service different from what he expects [22]. For example, if the requester requests a service for stock exchange quotes, and the provider returns a service supplying exchange rate quotes. This type of fault occurs if the description of a service is incorrect, or if the service provider misinterpreted the request from the requester [22]. QoS service failures occur during runtime [21].

Self-optimizing aims to select services at runtime, in order to maintain the expected QoS of the entire WSC [23]. The main objective of self-optimization WSC is to find the best Web service for each abstract service to achieve the FR as well as optimize QoS requirements. An abstract service is a set of Web service

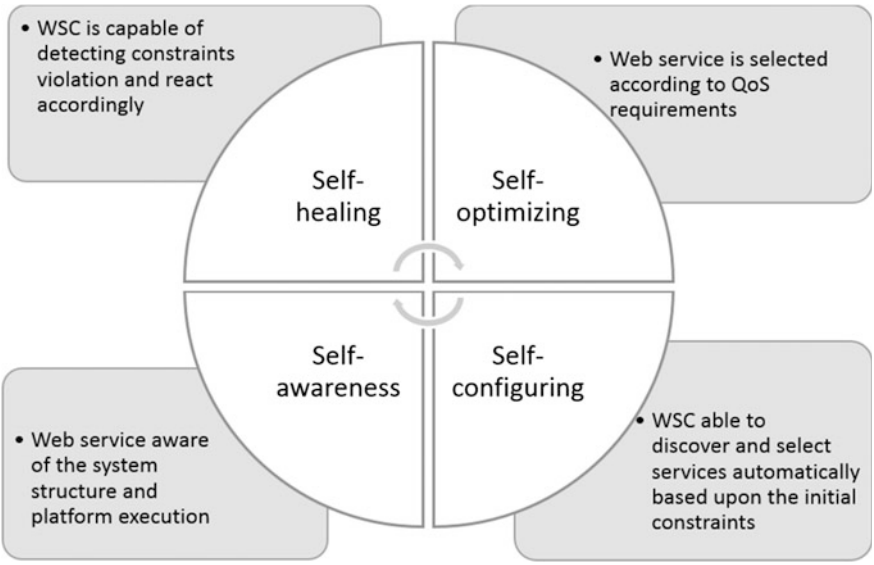

Fig. 10.2 Self -* properties in adaptive WSC

instance nodes in the WSC model that describes the functionality of the corresponding service [3]. Service selection for each abstract service is based on local or global QoS requirements. In the local optimization approach, service selection for each abstract service is based on the QoS of individual service. These approaches are useful in decentralized and dynamic environments. Local optimization approaches are the best in case there is no requirement to identify global constraints. This approach is suitable when the global QoS constraints are decomposed into local QoS constraints. The global optimization approach considers QoS constraints and preferences as a whole, e.g., when the whole response time is constrained.

Self-configuring aims to search for an optimal configuration of WSC components based upon the initial constraints [23]. Self-configuring WSC indicates that the WSC is able to discover and select services automatically. Self-awareness enables services to be aware of the system structure and platform execution. Self-awareness also enables the service to predict the impact of changes in their behavior and the effects of adaptation actions [14]. Self-awareness is aimed to ensure that the proactive adaptation of QoS requirements is satisfied [14].

10.3 RE for Self-Adaptive Systems

RE for dynamic adaptive systems is defined in the fourth level [10] as shown in Fig. 10.3. Level one is a general definition of the system and its reaction by developers. Level two is RE at runtime for achieving adaptation. Level three is

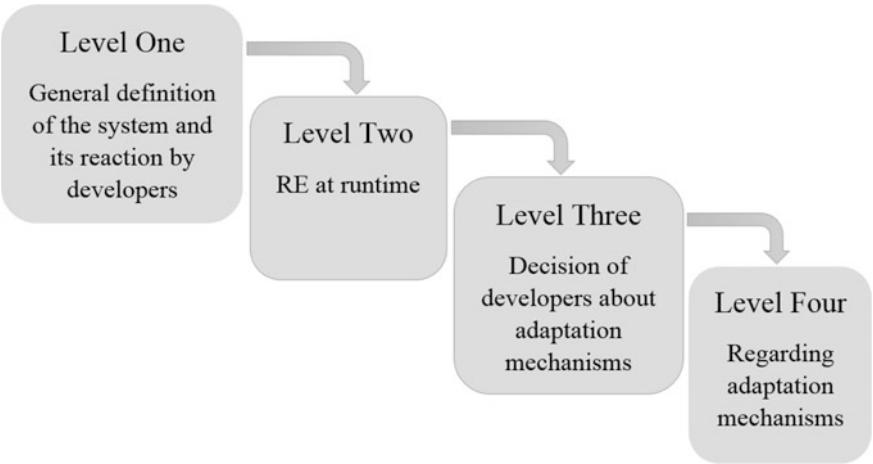

Fig. 10.3 Four levels to define RE for dynamic adaptive systems

decision of developers about adaptation mechanisms. Level four is research regarding adaptation mechanisms. RE for self-adaptive systems must deal with uncertainty because the execution environment information is unknown, and therefore the requirements for system behavior may need to change at run time in response to changes in the environment [24]. Requirement for self-adaptive system is specified as "incomplete" [24]. Chang [24] highlights research challenges for RE for self-adaptive systems. These challenges are new requirements language, mapping requirements language to architecture, managing uncertainty, requirements reflection, and traceability from requirements to implementation.

The traditional RE models such as i* and KAOS are not supported adaptivity or uncertainty. Various approaches are proposed to include runtime capabilities for RE. Baresi et al. [25] propose FLAGS, a goal model-based approach that generalizes the KAOS model, for modeling requirements at runtime. Pasquale et al. [26] present a FLAGS infrastructure to support requirements at runtime. Tropos 4AS is an agent-based methodology to model SAS requirement based on Tropos [27]. CARE is also modeling SAS requirement based on Tropos but it focuses on service-based applications for modeling [27].

The basic characteristics of the system become self-awareness and context-awareness to achieve adaptive behavior. Self-awareness describes the ability of a system to be aware of itself [10]. Context-awareness means that the system adapts its behavior based on the context of the application and the user [28]. Context is defined as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [10]. Bucchiarone et al. [29] propose a framework for adaptively of service-based applications according to context changing. This framework utilizes the concept of process fragments as a way to model processes. Business processes

and fragments are modeled as Adaptable Pervasive Flows (APFs). APFs add annotating activities with preconditions and effects besides classical workflow language. This makes business processes and fragments suitable for adaptation and execution in dynamic environments. At design time, abstract activities are specified for each fragment in terms of the goal it needs to achieve. Different adaptation mechanisms and strategies are used to handle dynamicity of context-aware pervasive systems. Adaptation mechanisms are refinement mechanism, local adaptation mechanism, and compensation mechanism. The adaptation strategies are one-shot adaptation, re-refinement strategy, and backward adaptation strategy; other context-aware approaches are founded in Sect. 10.5.2.

10.4 Requirements Specification Models in WSC

Li [6] classifies requirement models in WSC into three categories: WSC based on workflow, WSC based on Artificial Intelligent (AI) planning technique, and model-driven WSC as shown in Fig. 10.4. In WSC based on workflow, the logic of WSC can be captured using workflow pattern of Web service. In this approach, users' requirements are modeled in terms of workflow which refers to the logical execution order of action [6]. When implementing a WSC, atomic Web services are selected and invoked according to each action defined in the workflow, after that, the WSC is executed according to the predefined execution orders [6]. Workflow is generated in either a static or dynamic manner [30]. Static WSC workflow means that users are required to describe all the necessary actions and all possible execution orders among these actions. The selection of Web service is done automatically. In dynamic WSC workflow, creating business workflow or model and selecting Web service is done automatically.

WSC based on AI planning requires an algorithm to translate WSC problem to AI planning technique problem such as Planning Domain Definition Language (PDDL) [31], Hierarchical Task Network (HTN) [32], and graph plan [33]. This approach requires users to specify their composition requirements in different technical languages, which includes the descriptions of initial state, goal state,

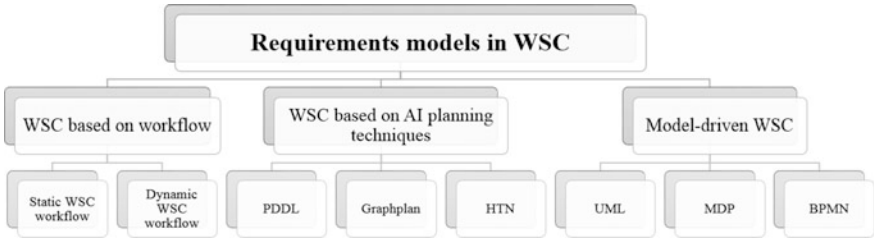

Fig. 10.4 Requirements models for WSC

possible domain states, and different actions that can be performed. For further information about WSC based on AI planning, it can be found in [34].

Model-driven approach for WSC uses models to describe user requirement (FR and NFR), business processes, abstract Web services, and dependence between Web services. The models are separated from executable WSC specifications. WSC can be modeled using Unified Modeling Language (UML) [35–37], MDP [38, 39], and Business Process Model and Notation (BPMN) [40, 41].

10.5 Classification of Self-Adaptive WSC Approaches

SAS modifies its own behavior in response to changes in the WSC environment. These environments are classified into the dynamic environment or static environment. In the dynamic environment, new WSC behavior and adaptation plans can be introduced during runtime. Contrary to the dynamic environment, the static environment is self-contained and not able to support the addition of new behaviors. In this section, approaches for reactive adaptation are classified into three categories: used of variability models, context-awareness, and multi-agent approaches.

10.5.1 Used of Variability Models

Variability is the ability of a service to change its behavior efficiently in the dynamic configurations [42]. The two important concepts concerning variability are variation points and variants. Variation points are located in a software system in which variation will occur, and variants are the alternatives that can be selected at those variation points [42]. Modeling and managing variability in a process can be classified into an architectural level and an implementation level. In an architectural level, variants are modeling inside software architecture such as BPMN and UML. In an implementation level, variants are modeling inside WSC language.

10.5.1.1 Architecture Level

Alférez et al. [41], Sun et al. [43], Yua et al. [40], and Xiao et al. [44] model variants in the architecture level to accommodate for changes that occur in FR. By changes in FR, we mean that the changes occur in a business logic or business requirement. Alférez et al. [41] create variability models and adaptation policies at design time to support the dynamic adaptive WSC. The composition model and variability model are separated. The dynamic adaptive WSC is described in adaptation policies in terms of the activation or deactivation of features in the variability model. The activation and deactivation of features in the variability model result in changes that occur in the WSC by adding or removing fragments of

Business Process Execution Language (WS-BPEL) code, that are deployed at runtime. The variability model and its possible configurations are verified at design time using Constraint Programming (CP). Sun et al. [43] extended ConIPF Variability Modeling Framework (COVAMOF) to allow it to configure the variability in a WSC. COVAMOF is a variability management framework that is used with software product families. COVAMOF variability concepts are modeled using UML diagrams and Variation point Interaction Diagram (VID). COVAMOF-VS tool suite is used to automated variability management in WSC at runtime.

Yua et al. [40] and Xiao et al. [44] propose model-driven based approaches for WSC. Yua et al. [40] propose an approach called the Model-Driven Development of Dynamically Adaptive Service-Oriented Systems with Aspects and Rules (MoDAR) to support the development of dynamically adaptive WS-BPEL-based systems. MoDAR includes the base model, the variable model, and the weave model. The base model follows the flow logic of the system. Variable model is used to take the decision aspect of a business requirement, which is changeable at runtime. Variable model is specified by business rule. Weave model is the aspect-oriented approach used to integrate the base model and the variable model. Xiao et al. [44] present model-driven variability-based WSC approach. Variability is defined within VxUML that is a UML extension. Class diagram, activity diagram, sequence diagram, and deployment diagram are extending to specify the variation points and variants. Variation point Interaction Diagram (VID) defines the dependencies between variation points and variants. Rule-based transformation language is used to transform VxUML to VxBPEL. VxBPEL is a BPEL extension to support variability at the implementation level.

Yua et al. [40], Alférez et al. [41], and Sun et al. [43] approaches are modeled to adapt the changes in the business process but these changes are fully known at design time to model the variability model. Furthermore, these approaches are not suitable for use in the dynamic environment because they are not able to deal with continuous and unanticipated changes in complex uncertain contexts.

10.5.1.2 Implementation Level

Imed et al. [45] solve the variability of QoS (vQoS) by introducing three variability operators: replicate, delete, and replace. Replicate and delete operators are used to adding and removing service instance in WSC, while the replace operator is used to change some faulty Web services. These operators are used to reconfigure automatic WSC when the SLA contract is violated. WSC reconfiguration (variability model) is modeling and verifying using Event-B. ProB model checker is used to trace possible design errors. Variability model is not required to define all at design time but variability operators that are used to adapt WSC are not enough to solve correctly vQoS problem. Koning et al. [46] propose VXBPEL language which is an extension of the standard BPEL language to adapt the changes in the business process. VXBPEL adds XML extension elements that store variability information inside the process definition BPEL which result in their being time-consuming,

tedious, difficult to manage, and error-prone. This approach is very complex in the case of having a large number of variation points. Furthermore, this approach is not working in the dynamic environment like those approaches at the architecture level. Sun et al. [47] also adapt the changes in the business process by executing VXBPEL WSC using VxBPEL ODE engine. The performance of VxBPEL_ODE is compared with VxBPEL_ActiveBPEL. From the experimental result, VxBPEL_ODE shows a comparable performance of VxBPEL_ActiveBPEL.

10.5.2 Context-Awareness

Alférez and Pelechano [48] present a runtime model to guide the dynamic evolution of context-aware WSC to deal with unforeseen QoS events in the dynamic environment. Tactics are used to preserve the requirements that can be negatively affected by unknown context events. These tactics are known at design time, but they are used to tackle unknown context events. The negative effect of selected tactics to other expected goals is not taken into consideration. Bucchiarone et al. [49] and Cubo et al. [50] focus on changes that occur in FR. Bucchiarone et al. [49] define a formal framework that uses a planning technique to adapt the execution of the WSC at runtime in case of context changes. At design time, the context properties and their evolution are modeled by defined context property diagrams. Context property diagrams present the possible values of the property as the diagram states and the changes of the property values as transitions. The changes of the service are annotated with the effects on the context properties. The business policy over the service is annotated with preconditions on the context property values to determine in which context setting the service may be executed. Adaptation activities are not explicitly represented inside context change. They are dynamically derived from the currently observed context, the state of a business process, and business goals. This framework is implemented and validated using a scenario from the logistics domain. Cubo et al. [50] extend Discovery, Adaptation and Monitoring of Context-Aware Services and Components (DAMASCo) framework with feature models to represent the variability and self-adaptive WSC according to context change situations. This approach is implemented in the Intelligent Transportation Systems (ITS) domain. This approach is not supported self-adaptive of the service to context change at runtime. This means that DAMASCo execution plan does not support the switching from one running configuration to another.

Li et al. [51], Cao et al. [52], and Wang and Tang [53] propose approaches that deal with changes that occur in FR and QoS. Li et al. [51] present case-based reasoning for self-healing ability in WSC. Previous failure instances as cases are stored in a case base. When a new fault occurs, the closest cases in the case base are retrieved. Cao et al. [52] present context-aware adaptive WSC framework that contains five main function modules. The first module is the design of BPEL process. The second module is the parse and execution of the BPEL document. The third module is the search agent. The fourth module is a context-aware agent. The fifth module is an update

agent. All of these modules are implemented using WSIG technology and Java language. The first three modules are used to execute BPEL process. The context-aware WSC is classified into service contexts and service composition contexts. Service context is responsible for gathering and checking context information before service establishment. Service composition contexts work while a composite service is performed. When perceiving the changes of contexts value, service composition may need to make some adjustment such as adding, deleting, or replacing a service, or fundamentally changing the whole combination process. When receiving a message about a variation of context value from a context-aware agent, update agent will search the most suitable policy from a policy library and send it to BPEL execution engine which will change the composition process according to chosen policy. Wang and Tang [53] present an architecture for self-adaption WSC. This architecture contains a context module that is responsible for adapting WSC to the changing at QoS and satisfies the service consumer's requirements. The context is categorized into service context, user context, and device context. Service context describes the properties of the service and the required execution environment of a service. These properties and preferences for services are written by a service provider and updated by user ratings. User context describes requirements and the environment that the service consumer can provide. Device context describes the real execution environment, including hardware and software environment. Changing contexts are handled according to user-defined personalized policies. Recomposition in Web services is made in a case where input and output changed only. Otherwise, changing contexts are handled according to user-defined personalized policies. This approach is not suitable for the dynamic environment because the contexts are predefined and other undefined contexts are not supported. It is difficult to predict all the possible situations arising in an uncertain environment.

10.5.3 Multi-Agent Approaches

Wang et al. [54] present self-adaptive WSC framework based on RL. MDP is used in this framework to model WSC. Workflows and alternative services are integrated into a single WSC. At runtime, the concrete workflows and services selection are specified based on the environment and the status of services. Q-Learning is used to find an optimal policy to follow up the dynamic environment. Wang et al. [55] extend the RL framework that was introduced in Wang et al. [54]. This study presents a Multi-Agent Reinforcement Learning (MARL) mechanism to enable adaptive WSC. The WSC process is modeled as MDP to adapt dynamic evolution of user requirements. The Q-learning algorithm is used to find an optimal policy to follow up the dynamic environment. This mechanism introduces a sharing strategy in the composition process to share information with an agent that make agent use the policies explored by the others. The MDP model needs complete knowledge and observation about the environment, which may be difficult to achieve in practical application. WSC may contain some failure services that can reach to a complete

disability of this WSC workflow. This case is not taken into consideration. Wang et al. [56] also proposed a new model for large-scale and adaptive WSC based on MARL. This model integrates State-Action-Reward-State-Action (SARSA) learning algorithm and same theory. Multi SARSA algorithm which is extended from single-agent SARSA is utilized to find the optimal solution. Team Markov Games (TMG) is used to model multi-agent WSC. This algorithm does not take into consideration the case of some failure service that can reach complete disability of this WSC workflow. Wang et al. [57] also use TMG to model multi-agent WSC like in Wang et al. [56] but it used Q-learning instead of multi SARSA algorithm.

Moustafa and Zhang [58] design two algorithms to fulfill data efficiency by saving experience data and using it to make updates to the learned policy. The first algorithm introduces an offline learning scheme for WSC. Offline learning scheme avoids the limitation of online reinforcement learning algorithms. This limitation is the time which is taken to achieve convergence which may exceed the limits imposed by service consumers. The second algorithm presents a coordination mechanism in order to enable MARL to learn the WSC task cooperatively. A collaborative learning algorithm is a group of independent agents who learn to organize their action selection strategies and each agent notifies other agents with its action selections to make WSC collaboratively. Q-table is used to connect and communicate with each agent directly. This shared Q-table records the most recent QoS information of Web services and the rate with which these services have been chosen by other agents. Hsieh and Lin [1] use Holonic Multi-agent System (HMS) architecture to design SAS systems. A Workflow Adaptation Problem (WAP) is formulated and an interaction mechanism between agents is proposed based on Contract Net Protocol (CNP) to find a WAP solutions. Self-* scheme is proposed to respond to the structural and non-structural change workflow. Structural changes refer to changes in FR, which means changes in a business process. Non-structural changes refer to changes in NFR such as changes in processing time, the number of available resources, and available time slots of resources. When the change occurs, an affected agent will apply CNP to determine the best services provided by the existing downstream agents.

10.6 Comparison and Limitations of Self-Adaptive WSC Approaches

In this section, we compare between the approaches we presented in Sect. 10.5 and present the limitations of some of these approaches. The comparison between these approaches is given in Table 10.1. We compare between these approaches according to

- Category of these approaches according to classification in Sect. 10.5;
- RE classification according to changes in FR, changes in QoS, and changes in both FR and QoS;

Table 10.1 Analysis and comparison between self-adaptive WSC approaches

Approaches	Category	RE classification	Adaptation mechanism	Composition model
Alferez et al. [41]	Used of variability models	FR	Feature model	BPMN
Sun et al. [43]	Used of variability models	FR	COVAMOF, UML and VID	BPEL
Yua et al. [40]	Used of variability models	FR	Business rule	BPMN
Xiao e1 al. [44]	Used of variability models	FR	VxUML	Not defined
Imed et al. [45]	Used of variability models	QoS	VxBPEL	Not defined
Koning et al. [46]	Used of variability models	FR	Event-B	Not defined
Sun et al. [47]	Used of variability models	FR	VxBPE	Not defined
Alferez and Pelechauo [48]	Context-awareness	QoS	Tactics strategies	BPMN
Bucchiarone et al. [49]	Context-awareness	FR	Planning techniques	Not defined
Cubo et al. [50]	Context-awareness	FR	Extend DAMASCo framework with feature models	BPEL and windows workflow foundation(WF)
Li et al. [51]	Context-awareness	FR and QoS	Case-based reasoning	WS-BPEL
Cao et al. [52]	Context-awareness	FR and QoS	Not defined	BPEL
Wang and Tang [53]	Context-awareness	FR and QoS	Personalized policies	SHOP2 as Al planning technique
Wang et al. [54]	Multi-agent approach	QoS	MARL	MDP
Wang et al. [55]	Multi-agent approach	QoS	MARL	TMG-WSC
Wang et al. [56]	Multi-agent approach	QoS	MARL	TMG-WSC
Wang et al. [57]	Multi-agent approach	QoS	RL	MDP
Moustafa and Zhang [58]	Multi-agent approach	QoS	RL MARL	MDP

- Adaptation mechanism which described in Sect. 10.2;
- Composition model which described in Sect. 10.4.

The limitations are summarized in Table 10.2. Most of the approaches, e.g., Alférez et al. [41], Sun et al. [47], Yua et al. [40], Xiao et al. [44], Imed et al. [45], Koning et al. [46], Wang and Tang [53], Wang et al. [54, [55], and Wang et al. [57], are not suitable for use in dynamic environments because they are not able to deal with continuous and unanticipated changes in complex uncertain contexts. A study by Imed et al. [45] is suitable for use in a dynamic environment through the use of variability operators, but they are not enough to solve correctly vQoS problem. Imed et al. [45], Koning et al. [46], and Sun et al. [47] store variability information

Table 10.2 Some WSC adaptation approach limitations

Approaches	Limitations
Alferez et al. [41], Sun et al. [43], Yua et al. [40] and Xiao et al. [44]	These approaches are not able to deal with continuous and unanticipated changes in complex uncertain contexts. This means that they are not suitable for use in dynamic environments
Imed et al. [45], Koning et al. [46] and Sun et al. [47]	These approaches store variability information inside languages which result in their being time-consuming, tedious, difficult to manage, and error-prone. They are very complex in the case of a large number of variation points. In addition, they do not work in dynamic environments
Alférez and pelechano [48]	The selected tactics strategies that are used to adapt the changes may be negative effective to other expected goals
Cubo et al. [50]	This approach is not supported self-adaptation of the service to context change at runtime. This means that DAMASCo execution plan does not support the switching from one running configuration to another
Imed et al. [45]	Variability operators that are used to adapt WSC are not enough to solve correctly the vQoS problem
Wang and Tang [53]	This approach is not suitable for the dynamic environment because the contexts are predefined and other undefined contexts are not supported. It is difficult to predict all the possible situations arising in an uncertain environment
Wang et al. [54, [57]	WSC process is modeled as a MDP model. This model needs complete knowledge and observation about environment, which may be difficult to achieve it in practical application
Wang et al. [54], [56, 57]	These algorithms are not taken into consideration the case of some failure service that can reach to a complete disability of this WSC workflow

inside language which result in their being time-consuming, tedious, difficult to manage, and error-prone. Therefore, their approaches are very complex in the case of a large number of variation points.

Alférez and Pelechano [48] use tactics to tackle unknown context events. The selected tactics strategies may be negative effective to other expected goals. Cubo et al. [50] approach does not support the switching from one running configuration to another at runtime. Wang et al. [54] and [55] model WSC process as a MDP model. WSC process is modeled as a MDP model. This model needs complete knowledge and observation about environment, which may be difficult to achieve in practical application. Wang et al. [54], [55], and Wang et al. [57] do not take into consideration the case of some failure service that can reach complete disability of this WSC workflow.

10.7 Conclusion and Future Work

WSC is a key issue in SOA. The objective of this chapter is to analyze and compare various self-adaptive WSC approaches to deal with the changes that may occur within and outside the dynamic composition environment. These approaches are classified into three categories: used of variability models, context-awareness, and multi-agent approaches. These approaches have some limitations. One of the limitations is that the approaches, which deal with changes that occur in QoS, adapt the WSC process based on changes in local and single QoS criteria. Another limitation is that most of these approaches are not able to deal with continuous and unanticipated changes in complex uncertain contexts because they need to define the contexts in design time and other undefined contexts are not supported. It is usually difficult to predict all of the possible situations that might arise in an uncertain environment. In future work, we intend to overcome these limitations by combining QoS-aware WSC approaches such as ant colony optimization or genetic algorithm with multi-agent approaches to obtain an optimal policy in case of multi QoS criteria. Partially Observable Markov Decision Process (POMDP) is used to model composition requirement instead of MDP because POMDP does not need the full knowledge observation of environment.

References

1. F.-S. Hsieh and J.-B. Lin, "A Self-adaptation Scheme for Workflow Management in Multi-agent Systems," *Journal of Intelligent Manufacturing,* vol. 27, no. 1, p. 131–148, 2016.
2. N. Ide and J. Pustejovsky, "What Does Interoperability Mean, Anyway? Toward an Operational Definition of Interoperability for Language Technology," in *Proceedings of the 2nd International Conference on Global Interoperability for Language Resources (ICGL)*, 2010.
3. B. Rohallah, M. Ramdane and S. Zaidi, "Agents and Owl-s based Semantic Web Service Discovery with User Preference Support," *International Journal of Web & Semantic Technology (IJWesT),* vol. 4, no. 2, pp. 57–75, April 2013.
4. I. sommerville, Software Engineering (9th Edition), 2011, p. 509.
5. L. Wang and J. Shen, "A Systematic Review of Bio-Inspired Service Concretization," *IEEE Transactions on Services Computing,* vol. PP, no. 99, p. 3, 2014.
6. W. Li, "Towards a Resilient Service Oriented Computing based on Ad-hoc Web Service Compositions in Dynamic Environments(Doctoral Dissertation)," *Institut d'Optique Graduate School,* 2014.
7. A. Moustafa and M. Zhang, "Towards Proactive Web Service Adaptation," in *Proceedings of the 24th International Conference Advanced Information Systems Engineering (CAiSE)*, 2012.
8. P. v. Eck and R. Wieringa, "Requirements Engineering for Service-Oriented Computing: A Position Paper," in *Proceedings of the 1st International Workshop on e-Services at ICEC*, 2003.
9. N. B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas and V. Issarny, "QoS-aware Service Composition in Dynamic Service Oriented Environments," in *Proceedings of the 10th International Middleware Conference*, 2009.

10. C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele and C. Becker, "A Survey on Engineering Approaches for Self-adaptive Systems," *Pervasive and Mobile Computing,* vol. 17, pp. 186, Part B, February 2015.
11. R. Aschoff and A. Zisman, "QoS-driven Proactive Adaptation of Service Composition," in *Proceedings of the 9th International Conference on Service Oriented Computing (ICSOC)*, 2011.
12. S. Vansyckel, D. Schäfer, G. Schiele and C. Becker, "Configuration Management for Proactive Adaptation in Pervasive Environments," in *Proceedings of the IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*, 2013.
13. C. Pahl, "Dynamic Adaptive Service Architecture—Towards Coordinated Service Composition," in *Proceedings of the 4th European Conference Software Architecture (ECSA)*, 2010.
14. S. D. G. Avila, "QoS Awareness and Adaptation in Service Composition(Doctoral Dissertation)," *The University of Leeds,* pp. 34–38, 2014.
15. D. Shuiguang, L. Huang, W. Tan and Z. Wu, "Top- Automatic Service Composition: A Parallel Method for Large-Scale Service Sets," *IEEE Transactions on Automation Science and Engineering,* vol. 11, no. 3, pp. 891–905, 2014.
16. J. Liu, J. Li, K. Liu and W. Wei, "A Hybrid Genetic and Particle Swarm Algorithm for Service Composition," in *Proceedings of the 6th International Conference on Advanced Language Processing and Web Information Technology (ALPIT)*, 2007.
17. Z. Shanshan, W. Lei, M. Lin and W. Zepeng, "An Improved Ant Colony Optimization Algorithm for QoS-aware Dynamic Web Service Composition," in *Proceedings of the International Conference on Industrial Control and Electronics Engineering*, 2012.
18. F. Qiqing, P. Xiaoming, L. Qinghua and H. Yahui, "A Global QoS Optimizing Web Services Selection Algorithm based on MOACO for Dynamic Web Service Composition," in *Proceedings of the 2009 International Forum on Information Technology and Applications*, 2009.
19. H. Raik, "Service Composition in Dynamic Environments: From Theory to Practice (Doctoral Dissertation)," *University of Trento,* p. 40, 2012.
20. S. Poonguzhali, R. Sunitha and G. Aghila, "Self-Healing in Dynamic Web Service Composition," *International Journal on Computer Science and Engineering (IJCSE),* vol. 3, no. 5, p. 2055, 2011.
21. S. Poonguzhali, L. JerlinRubini and S. Divya, "A Self-Healing Approach for Service Unavailability in Dynamic Web Service Composition," *International Journal of Computer Science and Information Technologies,* vol. 53, p. 4381, 2014.
22. K. May Chan, J. Bishop, J. Steyn, L. Baresi and S. Guinea, "A Fault Taxonomy for Web Service Composition," in *Proceedings of the International Conference on Service-Oriented Computing (ICSOC)*, 363–375.
23. S. D. G. Avila and K. Djemame, "A QoS Optimization Model for Service Composition," in *Proceedings of the 4th International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2012.
24. B. H. Cheng, R. d. Lemos, H. Giese, P. Inverardi and J. Magee, "Software Engineering for Self-Adaptive Systems: A Research Roadmap," in *Software Engineering for Self-Adaptive Systems*, 2009, pp. 1–26.
25. L. Baresi, L. Pasquale and P. Spoletini, "Fuzzy Goals for Requirements-driven Adaptation," in *Proceedings of the 18th IEEE International Requirements Engineering Conference*, 2010.
26. L. Pasquale, L. Baresi and B. Nuseibeh, "Towards Adaptive Systems through Requirements@Runtime," in *Proceedings of the 6th International Workshop on MODELS@Runtime*, 2011.
27. K. Angelopoulos, V. E. S. Souza and J. Pimentel, "Requirements and Architectural Approaches to Adaptive Software Systems: A Comparative Study," in *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2013.

28. G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," in *Proceedings of the 1st International Symposium Handheld and Ubiquitous Computing (HUC)*, 1999.
29. A. Bucchiarone, A. Marconi, M. Pistore and H. Raik, "Dynamic Adaptation of Fragment-based and Context-aware Business Processes," in *Proceedings of the IEEE 19th International Conference on Web Services*, 2012.
30. S. G. H. Tabatabaei, W. M. N. W. Kadir and S. Ibrahim, "A Review of Web Service Composition Approaches," in *Proceedings of the 1st International Conference on Computer Science and Information Technology (CCSIT)*, 2011.
31. O. Hatzi, D. Vrakas, M. Nikolaidou, N. Bassiliades, D. Anagnostopoulos and I. Vlahavas, "An Integrated Approach to Automated Semantic Web Service Composition through Planning," *IEEE Transactions on Services Computing,* vol. 5, no. 3, pp. 319–332, 2011.
32. X. Yihong, Z. Xianzhong and H. Xiaopeng, "Automated Semantic Web Service Composition Based on Enhanced HTN," in *Proceedings of the 5th IEEE International Symposium on Service Oriented System Engineering*, 2010.
33. Y. Bo and Q. Zheng, "Semantic Web Service Composition using Graphplan," in *Proceedings of the 4th IEEE Conference on Industrial Electronics and Applications*, 2009.
34. J. Rao and X. Su, "A Survey of Automated Web Service Composition Methods," in *Proceedings of the 1st International Conference on Semantic Web Services and Web Process Composition (SWSWPC)*, 2005.
35. B. Orriens, J. Yang and M. P. Papazoglou, "Model Driven Service Composition," in *Proceedings of the 1st International Conference Service-Oriented Computing (ICSOC)*, 2003.
36. Q. Z. Sheng and B. Benatallah, "ContextUML: A UML-Based Modeling Language for Model-driven Development of Context-aware Web Services," in *Proceedings of the International Conference on Mobile Business (ICMB)*, 2005.
37. C. C. Dumez, A. Nait-sidi-moh, J. Gaber and M. Wack, "Modeling and Specification of Web Services Composition using UML-S," in *Proceedings of the 4th International Conference on Next Generation Web Services Practices*, 15–20.
38. V. Uc-Cetina, F. Moo-Mena and R. Hernandez-Ucan, "Composition of Web Services using Markov Decision Processes and Dynamic Programming," *The Scientific World Journal,* vol. 2015, 2015.
39. A. Gao, D. Yangx, S. Tang and M. Zhang, "Web Service Composition using Markov Decision Processes," in *Proceedings of the 6th International Conference Advances in Web-Age Information Management (WAIM)*, 2005.
40. J. Yua, Q. Z. Shengb, J. K. Sweeb, J. Hanc, C. Liuc and T. H. Noorb, "Model-driven Development of Adaptive Web Service Processes with Aspects and Rules," *Journal of Computer and System Sciences,* vol. 81, no. 3, p. 533–552, May 2015.
41. G. Alférez, V. Pelechano, R. Mazo, C. Salinesi and D. Diazca, "Dynamic Adaptation of Service Compositions with Variability Models," *The Journal of Systems and Software,* vol. 91, pp. 24–47, 2014.
42. M. Svahnberg, J. v. Gurp and J. Bosch, "A Taxonomy of Variability Realization Techniques: Research Articles," *Journal of Software:Practice & Experience,* vol. 35, no. 8, pp. 705–754, 2005.
43. C.-A. Sun, R. Rossing and M. Sinnema, "Modeling and Managing the Variability of Web Service-based Systems," *The Journal of Systems and Software,* vol. 83, no. 3, p. 502–516, 2010.
44. H. Xiao, F. Yanmei, S. Chang-Ai, M. Zhiyi and S. Weizhong, "Towards Model-driven Variability-based Flexible Service Compositions," in *Proceedings of the IEEE 39th Annual International Computers, Software & Applications Conference (COMPSAC)*, 2015.
45. A. Imed, M. Graiet, S. Boubaker and N. B. Hadj-Alouane, "A Formal Approach for Verifying QoS Variability in Web Services Composition using EVENT-B," in *Proceedings of the 2015 IEEE International Conference on Web Services.*, New York, 2015.
46. M. Koning, C.-a. Sun and M. Sinnema, "VxBPEL: Supporting Variability for Web Services in BPEL," *Information and Software Technology,* vol. 51, no. 2, p. 258–269, 2009.

47. C.-A. Sun, P. Wang, X. Zhang and M. Aiello, "VxBPEL_ODE: A Variability Enhanced Service Composition Engine," in *Web Technologies and Applications*, 2014, pp. 69–81.
48. G. H. Alférez and V. Pelechano, "Facing Uncertianty in Web Service Compositions," in *Proceedings of the IEEE 20th Internatinal Conference on Web Services (ICWS)*, 2013.
49. A. Bucchiarone, R. Kazhamiakin, M. Pistore and H. Raik, "Adaptation of Service-based Business Processes by Context-aware Replanning," in *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2011.
50. J. Cubo, N. Gamez, L. Fuentes and E. Pimentel, "Composition and Self-Adaptation of Service-based Systems with Feature Models," in *Proceedings of the 13th International Conference on Software Reuse (ICSR)*, 2013.
51. G. Li, L. Liao, D. Song, J. Wang, F. Sun and G. Liang, "A Self-healing Framework for QoS-aware Web Service Composition via Case-Based Reasoning," in *Proceedings of the 15th Asia-Pacific Web Conference (APWeb)*, 2013.
52. Z. Cao, X. Zhang, W. Zhang, X. Xie, J. Shi and H. Xu, "A Context-aware Adaptive Web Service Composition Framework," in *Proceedings of the 2015 IEEE International Conference on Computational Intelligence & Communication Technology*, 2015.
53. B. Wang and X. Tang, "Designing a Self-adaptive and Context-aware Service Composition System," in *Proceedings of the IEEE Computers, Communications and IT Applications Conference (ComComAp)*, 2014.
54. H. Wang, Q. Wu, X. Chen, Q. Yu, Z. Zheng and A. Bougu, "Adaptive and Dynamic Service Composition Using Q-Learning," in *Proceedings of the 22nd International Conference on Tools with Artificial Intelligence*, 2010.
55. H. Wang, X. Wang, X. Hu, X. Zhang and M. Gu, "A Multi-Agent Reinforcement Learning Approach to Dynamic Service Composition," *Journal of Information Sciences,* vol. 363, pp. 96–119, 2016.
56. H. Wang, Q. Wu, X. Chen, Q. Yu, Z. Zheng and A. Bougu, "Integrating On-policy Reinforcement Learning with Multi-agent Techniques for Adaptive Service Composition," in *Proceedings of the 12th International Conference Service Oriented Computing (ICSOC)*, 2014.
57. H. Wang, Q. Wu, X. Chen, Q. Yu, Z. Zheng and A. Bougu, "Adaptive and Dynamic Service Composition via Multi-agent Reinforcement Learning," in *Proceedings of the IEEE International Conference on Web Services*, 2014.
58. A. Moustafa and M. Zhang, "Learning Efficient Compositions for QoS-aware Service Provisioning," in *Proceedings of the IEEE International Conference on Web Services*, 2014.