A Survey on QoS-Aware Evolutionary Web Service Composition

Fernando A. A. Lins, Nelson S. Rosa

Abstract— Currently, the use of Web services and service-oriented approaches to software development is a key topic in the computer science area. In addition, these services are being used in conjunction to build Web service compositions, which support the development and use of more complex systems through the combination of simpler functionalities. In this context, new services are being developed constantly, and these services may differ not only on its functional aspects (i.e., on its main functionalities), but also in non-functional aspects (e.g., performance and security). Considering this scenario, the service composition may evolve to a more suitable state (based on specific quality parameters) if the system is able to search for new services and evolve to another configuration (e.g., by switching services) that presents more advantages in comparison to the current status. This work presents a survey on the area of evolutionary Web service composition, in which evolution is guided based on QoS factors such as performance, security and cost. Current and relevant initiatives of these area were divided in three main dimensions, in order to better structure and present these initiatives.

Index Terms— evolution, web services, service composition, service-oriented computing (SOC), quality of service

1 Introduction

One of the most important topics of interest in the computing science, currently, is service composition [23]. The general motivation for this area is related to two main points: the possibility of producing more elaborate services from existing services and the possibility of business integration. In practical terms, the composition allows putting together services of different providers to create more sophisticated services. The importance of service composition has been recognized in the community because of its flexibility in building applications from primitive services in a plug-and-play form.

In this context, particularly with the advent of web services and the service-oriented computing (SOC) [23], new services are in constant development. In an application composed by various services, replacement may be needed due to diverse causes: unavailability, non-functional requirements (such as price, performance or security [18]) or even for version upgrade. One key point is based in the discovery and implementation of these changes in the shortest possible time. An evolutionary composition should be able to seek this new information, stay prepared for possible changes and to implement these changes without the need of rebooting systems (may cause loss of clients).

With also the advent of Cloud Computing, one important requirement that should be considered with respect to the SOC context is availability. Even with the use of physical contingence plans (available in cloud environments), a participant service may become unavailable due to various reasons (e.g. loss of connectivity or overload). One key point is how to ensure the correct continuity of the service composition. To preserve this continuity, the system should search for a new service and evolve to a new and valid configuration. Evolutionary mechanisms are even more important when services are spread across the Internet (and not only inside Clouds), based on the fact that the Internet is considered a more chaotic environment than the cloud.

Providing evolution/adaptability in a web service composition environment has been seen by the academic community as a complex task. For example, in the case of web service composition using WS-BPEL [19], a standard widely used in the development of business process, information about participant services must be obtained until the composition be deployed in the web server. This is carried out in a static way, in the source code of the application. However, it is also relevant to provide resources to enable the Web service compositions considering non-functional aspects such as performance, security and cost.

This paper focuses on the presentation of a survey composed by relevant initiatives in the area of evolutionary Web service composition. In order to structure and facilitate the understanding of the study, some important topics in the area were chosen to incorporate the search. Subjects related to "evolution", "adaptability", "reconfiguration", "automatic composition" and "on demand composition" were analyzed and are presented in next sections.

This paper is structured as follows: section 2 presents how this survey is structured. Section 3 introduces relevant works that focus on design issues. In the next section, 4, initiatives related to the instantiation and negotiation process in an evolutionary Web service composition are described. Section 5 focuses on works that propose evolutionary approaches and mechanisms to perform evolution at execution level. Finally, Section 6 presents the conclusions of this paper and highlights future work.

- *Fernando Antonio Aires Lins received his BS degree from the University of Pernambuco and MS/PhD degree from the Federal University of Pernambuco, Brazil. He is currently Professor at the Federal Rural University of Pernambuco, Brazil. His current research interests include service-oriented computing, non-functional requirements, business process management, cloud computing, automation and systems adaptability. E-mail: fernando.aires@deinfo.ufrpe.br.*
- *Nelson Souto Rosa received his BS, MS and PhD degrees in Computer Science from Federal University of Pernambuco, Recife, Brazil in 1992, 1996, 2001, respectively. Since 2001, he has been a Lecturer at Federal University of Pernambuco. His research interests include development of quality-aware middleware systems and service-oriented applications. E-mail: nsr@cin.ufpe.br.*

2 DIMENSIONS

In order to better classify the initiatives in the evolutionary Web service composition area, we introduce some criterions to better present them. These criterions are important because they classify all works in various "dimensions", which one composed of related works that have particular characteristics in common.

An intuitive way of defining dimensions for the evolutionary service composition is based on a traditional reference [7], which defines three stages to be examined in a evolutionary composition, based on the criteria "when":

1. **System architecture design.** In this level, the initial definition of functional and non-functional requirements (e.g., performance and cost) occurs. Requirements can be defined both at business level or process level.

2. **System negotiation and instantiation.** At this point, the composition instantiation will occur and the candidate Web services are searched, analyzed and selected. For this selection, activities such as negotiation agreements with partners and requirements checking at system level have a high importance. In this moment the initial negotiation between service providers and costumers takes place.

3. **System evolution.** When the system is running, it may be necessary that evolutions occur; based on that, renegotiations and new checks are carried out. In addition to adjust the composition itself, it is possible to occur an adaptation in the contract between clients and providers. Services may be replaced for a considerable number of reasons, and the composition should be prepared to this evolution, both at the technical level (application) and in the management level (contracts).

In the next sections, initiatives related to the design dimension, instantiation/negotiation dimension and the evolution dimension are analyzed. It is relevant to register that these initiatives may fit to more than one dimension (e.g., design and instantiation/negotiation). In these cases, they are classified based on where the main contribution of the initiative is located.

3 DESIGN DIMENSION

To develop an evolutionary web service composition, the first step is to model the functional and non-functional requirements (including those related to evolution). Considering the focus of this work, which is QoS-aware evolutionary Web service composition (i.e., the evolution of the service composition is based on quality elements/aspects), the modeling/specification of non-functional requirements becomes essential. This section presents strategies and related works that address this topic.

The first initiative focused on this section if Fluegge [4]. The author proposes the use of MDA [20] for services dynamic composition. A central idea of this pattern is that technology is very volatile, and there is a separation of platform-independent models (PIM) and models dependently on specific platforms (PSM), which has just providing facilities to executing compositions on a variety of technologies and adaptive mechanisms. The platform independent models (PIM) provide a high-level description of the system focusing only in the associated functionalities; the platform specific models (PSM) provides platform details related to the specification in the PIM model. Fluegge also proposes the utilization of computation independent models (CIM), which hides details about the structure of the model.

Figure 1 presents a composition process following the MDA-based approach proposed by Fluegge. A business user/developer specifies the CIM using a high-level tool or even manually. After that, a number of refinements between CIM, PIM, PSM and code are performed. Finally, the composition is deployed in an orchestration engine. The author also explains and defends the importance of incorporating the use of semantics to increase dynamism in service composition.

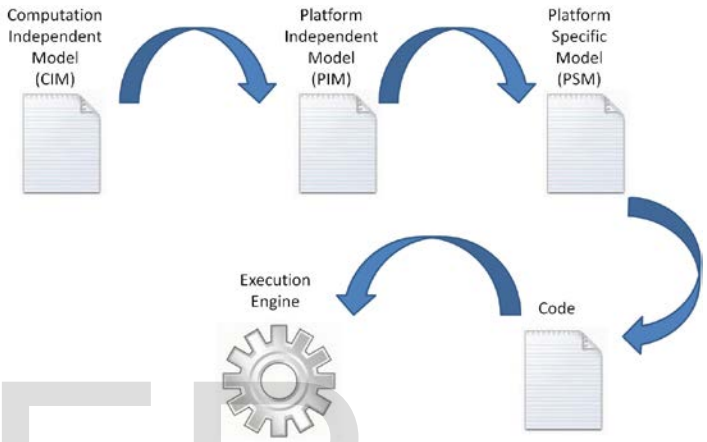


Fig. 1. MDA-based Service composition development process [4].

A relevant contribution of the Fluegge initiative is the proposition of using the concept of "abstract" services in the design/modeling level. By doing that, the service instantiation (focus of the next section) can occur during its execution (in this case, the service selection is delayed until the service is really needed), and not during the design of the service composition.

A relevant work that focus on non-functional requirements modeling is Sec-MoSC [12]. In this initiative, non-functional requirements are modeled through the use of four key concepts:

1. NF-Attribute. This concept/abstractions is used to represent both quantitative (e.g., performance) and qualitative (e.g., security) non-functional characteristics.

2. NF-Action. This abstractions represents an action that can impact in the quality of the NF-Attribute. For example, the NF-Action to use digital signatures is an action that have impact on the NF-Attribute Security.

3. NF-Statement. In general terms, represents a constraint that must be associated to a specific NF-Attribute. For example, the statement "high" can be associated to the NF-Attribute "Performance" to indicate that a high level of performance is desired.

4. NF-Property. To execute NF-Actions, generally more information (generally actions parameters) is needed. The NF-Property concept is proposed to represent this additional information that should be associated to a corresponding NF-Action.

The relationship of the aforementioned Sec-MoSC concepts

are illustrated on Figure 2. For example, a NF-Action may have 0 or more NF-Properties; however, a NF-Property must have one associated NF-Action, i.e., different NF-Properties may have the same name/value if they are associated to different NF-Actions. In addition, a NF-Attribute can be associated to 1 or more NF-Actions, and a NF-Action can be associated to one or more NF-Attributes.

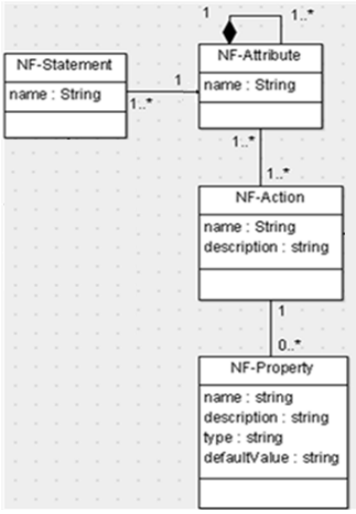


Fig. 2. Relationship of Sec-MoSC concepts. [12]

In addition to the aforementioned contributions, Sec-MoSC also proposes graphical notations (that can be used in service-based business processes modeled in BPMN [21]) to express non-functional requirements at high-level. This proposition facilitates the design of non-functional requirements in service compositions by high-level users (such as business experts, which generally does not have deep technical knowledge on how to implement services and associated quality attributes).

In turn, Gronmo [6] presents a strategy based on MDA for web service composition where semantic web (ontologies) and description of non-functional properties (such as price and security) are used for selection of services that meet a given composition. Initially is defined an abstract composition (activity diagram) that includes non-functional properties (classes diagram). This initial composition is transformed into a concret composition in BPEL. Instances of web services in the description BPEL are searched in repositories considering non-functional requirements defined in the composition. The web services are then ranked and selected for the composition implementation and execution.

Some parts of the Gronmo strategy can be performed automatically. Specially in the area of adaptive and evolutionary web service composition, this point is very important. It was concluded in this paper that the utilization of models helps the automation tasks. Additionally, the author says that the methodology provides a better documentation of the composition (based on the use of graphical models).

Another relevant work of the Design dimension is Schmeling [26]. The author proposes a model-driven approach to automate non-functional concerns (NFC) in composite Web services (i.e., NFC can be modeled and executed in service compositions using the Schmeling approach).

The proposed approach is basically composed by activities related to three main groups: functional concerns (FC) specification and realization, non-functional concerns (NFC) specification and non-functional concerns realization. Table 1, inspired on [26], presents the activities of this approach and their related groups.

TABLE 1
SCHEMELING APPROACH ACTIVITIES

Functional Specification and Realization	Non-Functional Specification	Non-Functional Realization
Definition of Process Logic	Process Requirements Specification	Action to Middleware Service Mapping
Identification & Selection of Partner Services	Requirements Alignment	Code Generation
Operation of Executable Process	Action Definition	
	Action Composition	
	Action to Service Mapping	

It is possible to note, initially, that the Schmeling approach proposes activities related to both functional and non-functional design of service compositions (modeled as business processes). In addition, these activities support not only the design phase, but also performs required translations and generates specific code for the execution of the service composition with the specified quality (e.g., using a security action such as cryptography). Finally, Schmeling approach was implemented in a tooling (the design of the service composition is carried out in a BPMN environment, whilst the runtime architecture is based on ESB) [26]. However, the author did not propose an element to specifically model non-functional properties.

One interesting initiative that appears in the context of MDA is Rodriguez [24]. Rodriguez extended the BPMN notation to allow the incorporation of security requirements into business process diagrams. BPMN is a standard widely adopted to express and execute Web service compositions; based on that, security requirements can be associated to service compositions. For each security requirement, a dedicated BPMN graphical notation is introduced and can be associated to various BPMN standard elements. This is done by the implementation of a BPMN security extension, which relates the BPMN standard graphical notations and the proposed security elements. Rodriguez introduces an intuitive graphical notation to represent security, which include internal codes to better specify distinct security requirements. However, such as Schmeling, this initiative does not focus on specifying non-functional action properties, such as cryptography algorithm and key length.

Menzel [15], in his work, presents a strategy based on MDA, proposing the use of domain-independent security models and security patterns. The aim of Menzel's work is to

facilitate the generation of security configurations based on the modeled requirements. For this purpose, it is used a model-driven approach: information at the modeling layer is gathered and translated to a domain-independent security model. Concrete protocols and security mechanisms are selected based on a security pattern. Models in the Menzel approach are also specified based on the Business Process Modeling Notation standard (BPMN). In the same way that Menzel translate high-level business security requirements to security resources at execution level, other non-functional properties, as evolution, can be also focused. This fact contributes to the adoption and dissemination of MDA-based approaches in evolutionary Web service compositions.

Finally, Lin [10] proposes a protocol for goal-based automatic/dynamic web service composition. The OWL technology is used as a basis for Lin's initiative. A language of goal description is proposed; this language attempts to shape the desires, preferences and requirements of users in service requisitions. Additionally, such language describes the targets to be met, relationships between these goals and restrictions to achieve the goals. As a final conclusion, Lin argues in favor of using a model-based methodology (and goal-oriented) to specify requirements in the context of automatic/adaptive web service composition.

Basically, Lin proposes a protocol stack composed by the following layers: Goal Description, Web Services Semantic Description, XML Message and Network Transfer Layer [10]. The first layer, Goal Description, is responsible for goal descriptions using an appropriate language. This language is used to specify goals to be achieved, relationships between the goals and so on. The next layer, Web Service Semantic Description, have the responsibility of describing the semantic meaning of each participant web service. The Web Service Interface Description Layer has information about the web service interfaces, specified mainly in WSDL, but having some further details specified in OWL-S. The last two layers, XML Message Layer and Network Transfer Layer, provide complimentary support to the execution of the web service composition.

4 Negotiation and Instantiation Dimension

After modeling the non-functional requirements, an important step of the evolutionary process is composed by the negotiation between service providers and clients and the instantiation (moment that candidates will be chosen to participate in the composition). Even if the user is able to choose a service at modeling level, the system can search if other possible alternative fits better the modeled functional and non-functional requirements (such as proposed by Fluegge [4]). This choosing process is important because the evolution may be guided based on the previous negotiations between server and clients. The focus of this section is how this negotiation may take place and how the services of the composition can be instantiated.

4.1 Service Negotiation

A key work in the context of service negotiation is presented by Han [7]. In a dynamic composition, activities such as selection of services and negotiation of non-functional properties have a high importance. The author suggests an approach with similar principles with SLA (Service Level Agreement). The important point to be noted is not only the approach itself, but the need (pointed out by the author) of a mechanism that allows the interaction ("agreements") between services in a standardized way. An approach, composed of 4 major activities, was proposed in Han's initiative. These activities are described as follows [7]:

1. **Selection of candidate services.** The first step in Han's approach is to select a set of candidate services that are able to satisfy the non-functional requirements desired by the user. At this moment, a high number or services may de discovered and the evolvable approach may maintain a list of them to be used in later stages (e.g. if a service fails, another alternative one may be used).
2. **Reaching service-level compatibility.** The negotiation process tries to exceed both side (client and server) expectations in order to satisfy the service-level requirements. Before the negotiation process, the candidate services should list its security properties in order to provide sufficient information to enable the selection process. To achieve this goal, Han proposes the use of automated negotiation agents; these agents eliminate services that do not have the desired non-functional requirements.
3. **Achieving system-level objectives.** This step is responsible to check if the services non-functional requirements matches the overall system-level objectives and requirements. Even if all chosen services have fulfilled individual non-functional requirements, there is no guarantee that the set of these services will achieve the system requirements. In fact, in a web service composition, it is not enough to state that a composition is secure based only in the fact that the participant services performs secure interactions. Model checking techniques may be used in this context.
4. **Creating security requirements.** After the negotiation and checking process, the final stage of the instantiation is to contract the selected services to be used in the composition. SLA may be used to specify all terms of the negotiation; the generated documentation can be used as a basis for future quality monitoring. The SLA documentation will be the legal and formal document that will represent the user requirements and it will represent also what the service is able to offer, including the expected quality.

Han also proposes contracts renegotiation; considering the SOC context, that services are created and changed frequently, renegotiating contracts may be important in an evolvable system; an evolution, as a consequence, may require some adaptation (in a service or in the composition itself). In this case, some SLA contracts must be reviewed. The author states that system evolution may result in one of the following scenarios: to adapt an existing service contract with different term and rules; to establish a new SLA contract with an alternative service; and to change the rules of the composition itself, that may lead to any of the previous scenarios. Finally, the author recognizes the importance of the non-functional requirement security in service composition suggesting a security-oriented framework for evolutionary composition of services. The focus

is the development of techniques in order to maintain or adapt service compositions in modified security contexts. In fact, security is now being considered one of the major non-functional requirement when dealing with web service composition.

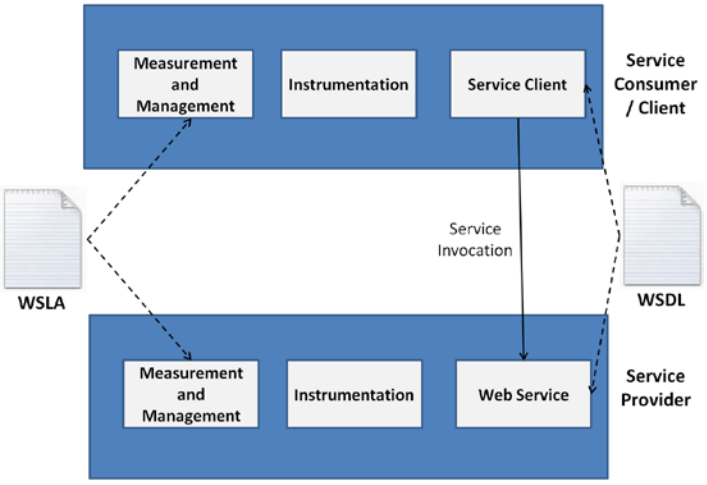
In turn, one of the most important initiatives in the area of negotiation between service clients and providers is WSLA (Web Service Level Agreement) [13]. WSLA can be defined as a framework to specify and monitore service-level agreements (SLA). In contrast to WSDL, that is focused on the functional aspects of the service, the WSLA specifies the non-functional characteristics desired by all involved parts and also specifies how to check and monitore them. WSLAs are agreements between a web service provider and a client and define the obligations of the involved parties. The service provider has the obligation to execute a service according to previously defined agreements based on IT-level service parameters (such as security, availability, performance and price).

Basically, WSLA is a language based on XML, and it is defined as an XML schema. XML is considered a key standard widely adopted in the web services context; this fact contributes to the proposition and dissemination of a higher-level XML-based language to express capabilities and requirements. Moreover, the WSLA standard can be used by both client and service provider to provide and supervise the web service quality. Client and provider may use the WSLA specification to monitore and to take appropriate actions by themselves. Clients can use third parties to make periodic checks.

In resume, the WSLA standard was proposed to describe and capture service-level agreements in order to enable adaptive/automatic configuration. Providers use it to implement the desired and negotiated quality as well clients may use it to check if the expected quality is achieved. To illustrate the proposal, Figure 3, inspired on [13], presents an overview of the relationship between service providers and clients in WSLA.

Fig. 3. WSLA architecture overview [13].

Initially, in can be noted in the figure that both service client and service provider have an "instrumentation" step; based on the WSLA specification, each party is able to check if everything is going as planned. Additionally, each one is able to access measured metrics from different sources; this allow clients to compare measured metrics with others vendors and organizations. It is also important to note that both service description (WSDL) and WSLA are accessible to providers and client. These two specifications provide, respectively, functional and non-functional information about the web service. Finally, the management is carried out by observing the WSLA specification and the instrumentation results.



The last work of this subsection (Negotiation) is the WDMPW (stands for WSLA Based Dynamic Monitoring and Pricing of Web Services) initiative [25]. The main contribution of this initiative is the proposition of a framework that dynamically measures and monitors QoS parameters based on WSLA assertions. Reports related to the violations of agreed quality levels are generated and sent to the corresponding users.

The general architecture of the WDMPW is based on a Third Party Broker, which is responsible for getting the metrics from the instrumentation function and for checking the performance agreed level specified on the WSLA assertion. Some internal services provides very interesting features, such as: Measurement Service (check the current performance of the service), Guarantee Level Check (implements evaluation functions), Reporting the Violation (both inserting the violation in a database and invoking a specific action in the interface) and Automatic Costing (the service usage cost can be calculated using this service) [25]. In short, the WDMPW initiative generates relevant information that can be used to negotiate and, especially, renegotiate contracts between service providers and consumers.

4.2 Service Instantiation

One interesting initiative in the area of service instantiation is Mei [14]. The author proposes an adaptive framework that bars problematic external services to be used in a service-oriented application. Dynamic WSDL information is available at public repositories and is used to support the analysis of popularly services at the moment. Following this idea, a service composition may be implemented by using highly referenced services. The work presents, as main contributions, two important points: a framework to select and compose services adaptively and an approach to perform services ranking using social network analysis.

In short, Mei's approach introduces some relevant resources, such as activities related to service registration, discovery and binding. When a SOC user wants to discover a service, this user should discover it in the repository using the discovery activity and, after than, bind it to the service through the use of the binding activity. Beside the important contribution of proposing a methodology to deal with adaptive service selection, that is considered a non-trivial task, this initiative does not concretely deal with how to specify the non-functional properties of the candidate services. This point is crucial, because the evolutionary system have to check (not only in the selection time, but also in execution time) the service properties in order to see if the desired quality is available and/or achieved.

Another interesting initiative in the service instantiation area is Han [8]. Han *et al* proposes a cloud service recommendation system to support the service selection process. The main objective of the work is to effectively recommend a good com-

bination of services to the consumers. The use of recommendation systems (RS) is to provide techniques to deal with the selection difficulties. The proposed service selection framework is based on the candidate services ranking; the user is now able to select the best candidates based on the associated rank.

The most relevant components of the architecture proposed by Han are: Web Portal, Application Specific Service, Request Manager, Resource Register and Resource Manager. The Web Portal is the interface of the system with the external world (specifically, users and cloud service providers). Users are able to access the recommendation system and to get the recommendation results; the cloud service providers may register their services into the system through the Web Portal. The Application Specific Services (SaaS) is divided into Hardware as a Service (HaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [8]. This is the traditional division of cloud services. The user will require one of these types of services depending of the nature of the application. The Request Manager is responsible to deal with the registration requisitions from the cloud service providers. After evaluating the requisition, the Request Manager passes it to the Resource Register, which is responsible to calculate the service rank and to store the result. All ranking results are stored in the Resource Repository, another element in the proposed recommendation system. The Resource Manager is in charge of controlling the resources of the cloud service providers.

The proposition of a recommendation system in the context of service selection is the main contribution of Han [8]. In fact, the idea of ranking the candidate services and presenting this result to the user is valuable and can be used in the context of evolutionary service composition. An interesting improvement that may be considered is the automatic selection of participant services; if the ranking results are available in the system, a composition may be automatically deployed based on these results, improving the autonomy of the system.

Another relevant reference in the area of service selection is Palanikkumar [22]. In this work, the author proposes the use of genetic algorithms and particle swarm optimization algorithm to the optimization problem of web service selection. A relevant contribution of this initiative is the proposition of relevant activities for the Web service selection in a QoS-based composition: 1) QoS modeling for concrete services; 2) computing the QoS of services composition, 3) fitness evaluation, 4) Web service selection based on genetic algorithm, 5) definition of chromosome, 6) genetic operations, 7) Web service selection based on PSO algorithm and so on. These activities are considered essential for an automated web service composition, especially considering evolutionary requirements, in which new services can arise eventually and the composition can increase its corresponding QoS attribute by the analysis of the new possibilities (e.g., changing a service for another one that is more secure or have better performance).

Batouche *et al* [1] also proposed an approach based on a genetic algorithm in order to optimize Web service compositions. This approach adopts OWL-S for the Web service description, and the reason is because this choice make the processing easier [1]. Basically, the proposed approach has the following activities: 1) services selection (service discovery is not focused, and the algorithm considers the existence of service registries), 2) to build the composition search space (set of possible solution) and 3) optimization by multi-objective evolutionary algorithms. The developed algorithm generates a set of possible (best) services to be used in the service composition. The author presents promising experimental results in the utilization of genetic algorithms in the evolutionary web service composition area (which includes, for example, analysis related to the fitting level of optimal solutions). Finally, it is important to highlight that the final decision is not of the approach/algorithm; the user is able to see the generated results and to select which ones should be considered.

5 Evolution Dimension

In the last two dimensions, two important and necessary steps were detailed; the modeling of the web service composition, including both functional and non-functional requirements, and the process of choosing services to participate in the composition. At this point, the focus is on how to use all of this information to enable the system evolution and how to provide and enforce evolutionary mechanisms.

In order to develop an evolutionary web service composition, a considerable number of aspects should be analyzed. For example, why should the system evolve? This evolution should be guided by functional requirements or by non-functional requirements? Which non-functional requirements should be observed in order to enable evolution desired by the user? The enforcement of the evolution is considered as one of the last steps to be performed, based on the reason that it depends of the user requirements (obtained in the design dimension), of the service selection and measurement and of the system execution itself (accommodating, in runtime, environment updates like service unavailability).

5.1 Aspect-oriented approaches

There is currently a considerable number of projects and researches on the web services composition area that are proposing the use of aspects to enable adaptability/evolvability [2][3][16]. This proposition allows the specification of general properties of the system being separated from the specification of its functionality. It can be stated that the use of aspect-oriented programming (AOP) brings significant contributions on the non-functional adaptation area, by providing languages for the expression of non-functional aspects of the system, such as competition, distribution, and exception handling [2]. This separated process of specification may facilitate the localization and adaptation of non-functional aspects. The main idea is to treat the adaptability as one aspect; aspects may be activated/deactivated at execution time. Basically, the aspect-oriented programming introduces modularity units called aspects. An aspect may contain code fragments ("advice") and location descriptors ("pointcuts") that inform where to insert these fragments.

Narendra [16] proposes an aspect-oriented solution to deal with adaptation in web service composition. More specifically, non-functional requirements changes guides the adaptation mechanisms in order to better satisfy the user desires. To insert this mechanism without using intrusive methods (such as

inserting more instructions in the application code), an aspect-oriented approach (AOP) is proposed. Moreover, Narendra uses a special type of aspect-oriented approach, distributed aspect-oriented approach, which is considered a non-intrusive mechanism, that can be better applied in the Web Services context. Narendra's aspect oriented solution was implemented on top of PROSE, an AOP implementation environment. This tool is based in the implementation of methods that intercept method calls in the JVM (Java Virtual Machine) in the exact point where the aspects should be executed.

In turn, Charfi [2] presents a solution based on middleware to support non-functional requirements in web service orquestrated compositions. This solution is based in process-based languages such as WS-BPEL (which does not have the proper support for such task). The composition and the middleware services are integrated through a BPEL-based aspects extension language named AO4BPEL. This extension enables dynamic changes and adaptability in the composition in a more modular way based on these non-functional requirements, since midflight adaptations can be implemented with the use of advices in AO4BPEL. Based on the idea that a business process in BPEL consists in a set of activities, and taking into account that these activities can be considered well-defined points in the execution of processes (known as join points), each BPEL activity becomes a possible join point. The attributes of the web service composition (or the attributes of a particular service) can be used as a basis for the choice of relevant join points. Charfi also proposes a syntactic change in the WS-BPEL standard, inserting the tag <pointcut>, responsible for selecting the activities where the execution of specific functionalities will be integrated.

The architecture of Charfi's solution is based on the main components BPEL Runtime and the Aspect Manager [2]. The BPEL Runtime can be considered as an extended process interpreter (supporting the new tags added due to the addition of aspects in the project). This component is also responsible for managing the processes instances and the aspects. The component Aspect Manager controls the aspects execution. Other relevant component of the architecture is the Aspect Definition and Deployment Toll, which manages the registration and activation/deactivation of aspects. It uses an aspect repository, place where aspects should be stored and previously implemented.

It is important to note that Charfi guided the initiative based on the known limitations of language/patterns for web service compositions (for example, WS-BPEL suffers from a lack of adaptive/evolutional support), and the proposed approach minimizes the deficiency of this particular issue. However, proposing an extension that changes the traditional and well-known WS-BPEL standard syntax (by adding a new element <pointcut>) may imposes difficulties in the adoption and dissemination of the initiative.

5.2 Other Approaches

In a relevant initiative, Fujii [5] proposes an approach to dynamically compose distributed components (services) on demand, with the construction of new services (new compositions) without needing pre-installation. Fujii proposes the framework CoSMoS (Component Service Models with Semantics), which is a semantic-oriented model of components. In CosMoS, a component is described by its operations and its properties. Moreover, this architecture also introduces concepts, entities that represent abstract semantic ideas (e.g. directions), and actions. Semantic annotations of operations, inputs, outputs and properties of the services are also made using concepts. To support the CosMoS environment requirements, a middleware called CoRE was implemented [5]. CoRE has, basically, two interfaces, the Discovery Interface, that provides a way to search from components in a network, and the Access Interface, that provides a unique manner to invoke a component operation and to retrieve a component property.

In turn, Nie [17] proposes an approach to support on demand services composition. A language for service composition is proposed in order to describe user requirements. This description is made using a combination of semantic aspects and SLA (Service Level Agreement). Additionally, this approach supports the link between participants of the composition at run-time, in a dynamic way. One of the main contributions of the initiative is the proposition of a Composition Broker [17], which executes composite services based on business description with automatic discovery of services. This element (composition broker) has a central role throughout the architecture: the description of the composite service is analyzed (analyze) and then the function of discovery (discover) is used to dynamically find Web services based on user demands. The broker is also responsible for making the call (invoke) and receiving the answer (reply) from the chosen service providers.

The main focus of this architecture is how to dynamically discover services based on the users demand and how to provide an automatic implementation of the composite service. For this purpose, formal semantics and SLA are used by the author. Semantics descriptions are used primarily to make the matching of services features based on its semantic and the SLA is applied when dealing with issues related to QoS between the client and the provider of the service.

Lins [11] presents an alternative approach for adaptive service composition with specific focus on the exchange level of partners and services at runtime. This idea is based fundamentally in changing the semantics of one primitive of the WS-BPEL, <invoke>. This primitive is responsible for the invocation of web services within the composition. In the current context, the invoke primitive executes on a static way: the name of the partner (entity that provides the web service) and the parameters for the service implementation are statically described in the source code. If any problem occurs with this partner, the application stops its operation, becoming unavailable until necessary changes are carried out. Through Lins proposal, the invoke primitive acquires a dynamic behavior, in which it becomes able to verify if there is any change in the environment and to act in case further actions are required or, at least, recommended.

To enable this modification in the WS-BPEL invoke semantics, Lins proposed an extension to an execution engine that supports WS-BPEL (more specifically, ActiveBPEL). This extension was named A-ActiveBPEL (stands for Adaptive ActiveBPEL). With the incorporation of this adaptable extension

to the original engine, this engine then acquire an adaptive behavior. This change does not alter the syntactical specification of WS-BPEL in applications; what changes is the implementation of the engine, which presents an adaptive behavior, completely transparent to programmers. Figure 4 presents an overview of the A-ActiveBPEL architecture. The syntax of the invoke primitive remains unchangeable in the source code of the application, but the extension defines a new semantics to this primitive. Before calling participants web services, the invoke primitive checks if there are changes in the environment (e.g., new services or even updates in the services quality). To support this, two functions are used; adaptiveInvoke (to call the adaptive behavior) and informWS (to return the selected service). The extension checks in the Web Service Repository if any event has occurred and if any action is required. Next, the primitive forwards the request to the appropriate web service.

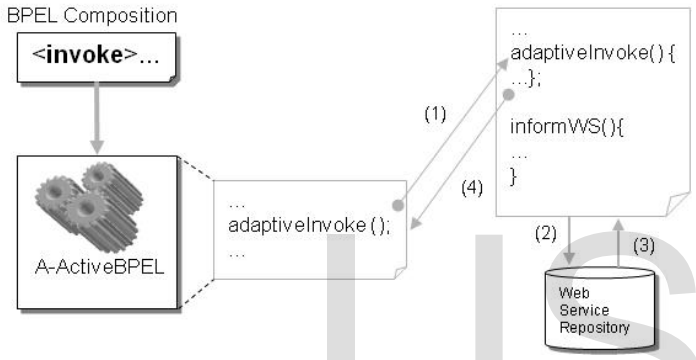


Fig. 4. A-ActiveBPEL Architecture [11].

In turn, Karastoyanova [9] presents a proposal to perform the adaptation of composed web services at deployment time. This proposal consists of an extension to the standard WS-BPEL which implements an adaptive mechanism called "find and bind". This mechanism facilitates the control of the services selection and also provides a process instance repair mechanism. To implement this mechanism a modification was proposed in the WS-BPEL syntax, with the addition of a specific tag called <find_bind>. In other words, the developer should, explicitly in the application source code, specify in what point can occur the exchange of web services and which are the preferred candidates to be selected by the adaptive extension.

Karastoyanova approach is basically composed by four steps: search, selection, bind and invocation. In every occurrence of the primitive <find_bind> in the source code, these four steps are performed. Firstly, the extension performs a search to find candidate web services. This search may include various repositories (including UDDI, local and proprietary repositories). The second step performs the selection of which web services should be used in the composition. This choice is based on diverse criterions; the user can define which criteria should be used. The next step is the binding; the extension perform implementation tasks to build the composition with the new configuration. Finally, the selected web services are invoked to execute the required business task. One limitation of this initiative is related to the fact that the adaptability occurs only in deployment time, not in execution time.

Finally, Xie [28] proposes another relevant work in the area of QoS-driven web services evolution, which aims to enable the evolution of Web services in dynamic environments with the prediction of future service QoS values using real-world QoS dataset. Considering that service compositions are basically carried out by a set of Web services, it is possible to use Xie's approach to predict future QoS properties of the composition services, which can be used to infer QoS useful information about the composition itself. It is possible to state that Xie's approach can be used in conjunction with other evolutionary techniques previously presented in this survey to provide additional information related to the service composition evolution (by informing the QoS predictions related to the participant services).

6 Conclusions and Future Work

Service oriented computing (SOC) is playing an important role in the computer science area. In the SOC context, services may be developed everyday, in the same way that a widely used service may be discontinued or even become unavailable in a specific period of time. To deal with such dynamic environment, diverse initiatives are proposing methodologies and extensions to enable adaptability/evolvability in Web service composition. This work presents a survey of this area, and considerations related to current initiatives were performed. These works were categorized into three dimensions – Design, Negotiation/Instantiation and Evolution. It is important to note that all three dimensions were contemplated with a considerable number of works; this fact shows that the proposed division in these three dimensions was valid and the dimensions itself (design, negotiation/instantiation and evolution) were correctly chosen.

As future work, we intend to use the presented initiatives as a basis to a more complete solution to deal with evolution, starting from the high-level modeling (the first dimension) to the execution, monitoring and evolution of the composition (related to the last dimension).

References

[1] Batouche, B., Naudet, Y. and Guinand, F. "Semantic Web Services Composition Optimized by Multi-Objective Evolutionary Algorithms", in Fifth International Conference on Internet and Web Applications and Services, pp. 180-185, 2010.

[2] Charfi, A. and Mezini, M. "Using Aspects for Security Engineering of Web Service Compositions", in: Proceedings of the 3th IEEE International Conference on Web Services (ICWS), pp. 59-66.

[3] Charfi, A. and Mezini, M. "Aspect-Oriented Business Process Modeling with AO4BPMN", in 6th European Conference on Modeling Foundations and Applications (ECMFA 2010), 2010.

[4] Fluegge, M.; Santos, I. J. G.; Tizzo, N. P. and Madeira, E. R. M. "Challenges and techniques on the road to dynamically compose web services", in Proceedings of the 6th international conference on Web engineering', ACM, New York, NY, USA, pp. 40-47.

[5] Fujii, K. and Suda, T. "Semantics-based dynamic service composition", in IEEE Journal on Selected Areas in Communications, vol. 23,

no. 12, pp. 2361-2372.

[6] Gronmo, R. and Jaeger, M. C. "Model-Driven Semantic Web Service Composition", in 'Proceedings of the 12th Asia-Pacific Software Engineering Conference', IEEE Computer Society, Washington, DC, USA, pp. 79--86.

[7] Han, J. *et al*. Security-Oriented Service Composition and Evolution, in IEEE XIII Asia Pacific Software Engineering Conference (APSEC'06). C. J. Kaufman, Rocky Mountain Research Laboratories, Boulder, Co 2006.

[8] Han, S., Hassan, M. M., Yoon, C. and Huh, E. "Efficient Service Recommendation System for Cloud Computing Market", in Proceedings of the 2nd International Conference on Interaction Sciences (ICIS 2009). Seoul, Korea. 2009.

[9] Karastoyanova, D. *et al*. "Extending BPEL for Run Time Adaptability", in: Proceedings of the Ninth IEEE International Enterprise Computing Conference, pp. 15-26.

[10] Lin, M.; Guo, H. and Yin, J. " Goal Description Language for Semantic Web Service Automatic Composition", in 'Proceedings of the 2005 Symposium on Applications and the Internet (SAINT'05)', IEEE Computer Society, pp. 1-7, 2005.

[11] Lins, F.A.A., Junior, J.C.S. and Rosa, N.S. "Adaptive Web Service Composition", in SIGSOFT Software Engineering Notes, 32(4), 2007.

[12] Lins, F.A.A. *et al*. "Towards an Approach to Design and Enforce Security in Web Service Composition", in International Journal of Web Engineering and Technology, pp. 323-357, 2012.

[13] Ludwig, H. *et al*. "Web Service Level Agreement (WSLA) Language Specification". IBM Corporation technical report version 1.0.

[14] Mei, Lijun; Chan, W.K. and Tse, T.H. "An Adaptive Service Selection Approach to Service Composition", in Proceedings of IEEE 6th International Conference on Web Services (ICSW 2008), 2008.

[15] Menzel, M., Homas, I. and Meinel, C. "Security Requirements Specification in Service-Oriented Business Process Management". In International Conference on Availability, Reliability and Security. Fukuoka, Japan. 2010.

[16] Narendra, N. *et al*. "Run-time Adaptation of Non-functional Properties of Composite Web Services Using Aspect-Oriented Programming", in Proceedings of the International Conference on Service-Oriented Computing (ICSOC 2007). 2007.

[17] Nie, T. *et al*. "An Approach for Composing Web Services on Demand", in Proceedings of 10th International Conference on Computer Supported Cooperative Work in Design (CSCWD '06'), pp. 1-6.

[18] OASIS. "Web Services Quality Factors version 1.0", technical report (candidate OASIS standard). Available at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsqm (last visit: 02th August 2014). 2012.

[19] OASIS. "Web Services Business Process Execution Language". Available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel (last visit: 5th June 2014).

[20] OMG. "A Proposal for a MDA Foundation Model". Technical Report, OMG. 2005.

[21] OMG. Business Process Model and Notation v 2.0. Available at http://www.omg.org/spec/BPMN/2.0/PDF/ (last visit: 24th Feb 2014).

[22] Palanikkumar, D. and Kousalya, G. "An Evolutionary Algorithmic Approach Based Optimal Web Service Selection for Composition with Quality of Service", in Journal of Computer Science, v. 8(4), pp, 573-578, 2012.

[23] Papazoglou, O. *et al*. "Service-oriented Computing: State of the art and research challenges", in IEEE Computer, 40(11), pp. 38-45, 2007.

[24] Rodriguez, A., Fernandez-Medina, E. Trujillo, J. and Piattini, M. Secure business process model specifications through an UML 2.0 activity diagram profile. Decision support systems, v. 51, pp. 446-465. 2012.

[25] Sha, M. *et al*. " WSLA Based Dynamic Monitoring and Pricing of Web Services", in International Journal of Scientific & Engineering Research (IJSER). v. 4(2), pp. 1-6. 2013.

[26] Schmeling, B. *et al*. "Composing Non-Functional Concerns in Composite Web Services", in IEEE International Conference on Web Services, pp. 331-338, 2011.

[27] Webster, D., Townend, P. and Xu, J. " Restructuring Web Service Interfaces to support Evolution", in 2014 IEEE 8th International Symposium on Service Oriented System Engineering, pp. 158-159, 2014.

[28] Xie, Q., Wu, K. and Xu, J. "Qos Driven Web Services Evolution", in 2011 International Conference on Complex, Intelligent, and Software Intensive Systems, pp. 329-334, 2011.