ISSN 2277-3061

SOADM: A Design Architectural Method for Service-as-a-Software

Hamid Mcheick, Youcef Baghdadi
Computer Science Department, Université du Québec à Chicoutimi, Canada
Hamid_Mcheick@uqac.ca
Department of Computer Science, Sultan Qaboos University, Muscat, Oman
ybaghdadi@squ.edu.om

ABSTRACT

Service-Oriented Software Engineering is a new approach to build software solutions as services and compositions with respect to service orientation and service-oriented architecture (SOA). SOA provides the methodology to form applications (web services) as functional building blocks to build an accessible infrastructure for consumers, which is autonomous and not platform-dependent. Platform independence makes it easier to develop web services, such as operating systems, programming languages, etc. Several methods from both academia and industry have been developed for service-oriented based systems. This work first questions “to what extent a solution provided by a method would conform to service orientation, particularly, how to examine the design decisions based on quality attributes”, and “to what extent the method would align solutions with problems”. Next, it shapes methods with a framework. Then, it proposes SOADM, a method for developing Service-as-a Software (SaaS) in high level design based on functional requirements and quality attributes.

Indexing terms/Keywords

Web services; Software Architecture; Service Orientation; SOA; SaaS; Framework for Development Methods; SOADM method

Academic Discipline and Sub-Disciplines

Computer Science, Information Systems

SUBJECT CLASSIFICATION

Methodology, Framework

TYPE (METHOD/APPROACH)

Literary Analysis; Research Design

Council for Innovative Research

Peer Review Research Publishing System

Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY

Vol 12, No. 10

editor@cirworld.com

www.cirworld.com, www.ijctonline.com

INTRODUCTION

Developing SOA-complaint solutions requires a new engineering approach that encompasses three perspectives: (P1) service, (P2) composition, and (P3) management (Papazoglou *et al.*, 2011). The service perspective concerns with basic service identification, development, test, and deployment. The composition perspective concerns with mechanisms such as orchestration and choreography, used to compose basic services into software solutions. The management perspective deals with the inspection and quality of the deployed services and compositions.

This new service-oriented engineering approach, referred to as Service-Oriented Software Engineering (SOSE), would certainly sketch out methods (including processes, models, languages, formalisms, tools, notations, and patterns) that embody the three perspectives. As software community has facing challenges in this area. A number of methods, for SaaS, from both academia and industry have been compared (Al Rawahi and Baghdadi, 2005; Baghdadi, 2012a; Brittenham, 2001; Kohlborn *et al.*, 2009; Gu and Lago, 2010; Gu and Lago, 2011). This results in two main findings:

F1. Each of the existing method has considered limited perspectives of SOSE (Gu and Lago, 2011)

F2. Traditional top-down and bottom-up approaches need to be extended to green-field and meet-in-the-middle (Baghdadi, 2013; Chen and He, 2011), specifically to deal with legacy applications

First, this work considers the following set of sound perspectives as a kind of framework that guides the engineering of methods for SaaS:

1. SO design principles and SOA principles and drivers
2. Business/IT alignment that considers different levels of abstraction and refinement, namely business and technology building blocks, we refer to as Business-Oriented Building Blocks (BOBs) and Technology-Oriented Building Blocks (TOBs) respectively to systematically move from requirements to solution.
3. Software engineering challenges, process models, and CASE tools supporting both phases of service development and composition (Baghdadi, 2014)
4. Inspection of solution quality attributes and design decisions. Indeed, quality attributes can drive and determine the architecture design (Bass, *et al.*, 2003).

Next, it questions:

Q1] to what extent a solution provided by a method would conform to SO and SOA, particularly, how to examine the design decisions based on quality attributes?

Q2] to what extent a method aligns solutions with problems?

Then, a framework is conceived to: (i) answer these two questions, (ii) define the peculiarities of SOSE methods, and (iii) guide a new method; we refer to as Service-Oriented Analysis and Design Method (SOADM).

The remainder of the paper is organized as follows: Section 2 summarizes the framework for SOA-based solutions. Section 3 compares the different types of service development methods within the framework. Section 4 presents SOADM, a newer approach to develop services. Section 5 presents a case study to illustrate SOADM method. Some related work is described in section 6. Finally, a conclusion section presents further development and open issues.

FRAMEWORK: GUIDANCE TOWARDS A METHOD FOR SAAS

The framework aims at sketching out the elements of SOSE in order to guide the engineering of SOSE methods that will be used for developing services that comply with SO and SOA. It helps answering the following questions:

Q1] to what extent an approach considers critical perspectives that are: (i) SO designs principles, (ii) SOA principles and drivers, (iv) solution life cycle models and tools, and (v) inspection parameters for quality of the solutions provided by the type of the methods, particularly, how to examine the design decisions based on quality attributes?

Q2] to what extent an approach aligns solutions with problems by considering services as building blocks at both business and IT levels. That how an approach expresses the elements of the reality and their mapping into services

The framework consists of a set of sound perspectives a developer of a method for SaaS needs to consider:

2.1 Business-Oriented Building Blocks

To align IT to business (Baker *et al.*, 2010), a method should consider how to move systematically from business requirements to IT solution. This is possible only if the building blocks, core elements upon which a solution is built, are considered at different levels of abstraction so that various types of stakeholders have their restive, understandable view of the solution. This should be made possible by the techniques of refinements of different abstractions.

The general objective of service mapping is to identify services that are valuable, either from business or IT perspective. However, what constitutes a service has different meanings due to the lack of a standard classification framework of identified services. Several researchers attempted to provide classification of services (Gu and Lago, 2010; Bell, 2008; Cohen, 2007). Four kinds of services are described in the litterature:

1. Conceptual services (also called business services) service represents the core of software product requirements. They express organizational ideas, thoughts, opinions, views, or themes that propose software solutions to

organization problems (Bell, 2008). A conceptual service is a specific set of actions that are performed by an organization (Marks and Bell, 2007). From the perspective of business value, a conceptual service creates or captures value via the provider/consumer interaction (Kohlborn et al., 2009).

2. Realization services (aka software/IT service) describe part of an application system, which can be consumed separately by several entities (Bell, 2008). It realizes some function and can be invoked by other services or components.
3. Infrastructure services are part of the organization supporting distributed computing infrastructure (Cohen, 2007). These are communication services and auxiliary services.
4. Capability services provide an explicit business value, ranging from generic services that are reused in multiple service-oriented applications to specialized services that are part of a single service-oriented application. Capability services include:
   - Process services are services whose operations are directed by the BP definition. A composition of other services that provide specific sets of functions to execute BP logic (Erl, 2008; Cohen, 2007) and workflows that realize them. BPEL are example of such kinds of services.
   - Task services (aka application/activity/capability service) encapsulate business logic specific to activities or BP that are useful across business units. A task service represents an atomic unit of process logic.
   - Entity services (also called data service) represent one or more related business entities (documents in a business model) for instance, customer or invoice. Entity service is considered a highly reusable service because it is agnostic to most process services representing BP and workflow (Erl, 2008).
   - Utility services contain logic derived from a solution or technology platform. Utility services expose capabilities of multiple applications and resources within the organization including migrated, reengineered legacy systems, and some Commercial Of-The-Shelf software (COTS).
   - Hybrid services contain logic derived from both BPs and applications. Hybrid services expose capabilities wrapped legacy systems and some COTS that may exists within the organization. One reason hybrid services exists is that some legacy systems cannot be easily converted to types of services other than hybrid.
   - Partner services are offered to/by external partners based on agreed terms, this type of services is known as Application as a Service (AaaS) in Cloud based solutions (Patterson, 2012). Partner services are considered as separate type due to security and management concerns. A partner service could present capability offered by any of the application services depending on the organizational requirements.

In general, services can be divided into two broad categories BOBs and TOBs. This work considers breifly these two categories.

2.1.1 BOBs

BOBs are building blocks used at higher abstraction levels by the stakeholders. Indeed, building blocks identified at a higher level of abstraction allow creation of reusable, well documented and independent of technology/platform solutions (Frantz, 2008). The purpose is to align IT solution to business.

2.1.2 TOBs

Most of existing methods for developing WSs are based on lower abstraction levels; we refer to as technology-oriented buildings blocks. These building blocks are used at design time such as Entity Types in E~R model, UML class diagram, UML use cases; or at run-time such as any piece of code (e.g. legacy applications, stored procedures, Java beans, or EJB).

2.2 Service Orientation and SOA

Understanding Service Orientation (SO) and Service Oriented Architecture is critical for SOSE methods.

2.2.1 SO

SO is defined in (Erl, 2008) as "*a design paradigm comprised of a set of design principles, whereby a design principle is* an accepted industry practice". In this definition, the service is the most fundamental unit for building service-oriented solutions such as service-oriented software (SOS).

To comply with SO and to be a piece of composition, a service should exhibit some desirable properties that enforce the loose coupling and interoperability in addition to the well-known principles of separation of concerns and information hiding. These desirable properties are: Self-containment, Autonomy, Abstraction, Standard Contract, Granularity, Stateless, and Dynamic binding.

2.2.2 SOA

OASIS (OASIS, 2006) defines SOA as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations".

Actually, SOA is originally an architectural approach for business (Cummins, 2009; Cater, 2011). However, it may also be considered as computational architecture (Bean, 2010; Davies *et al.*, 2008; Erl, 2008) as it may be seen as an evolution of the component-based engineering and distributed computing paradigms and architectures such as RPC, CORBA, DCOM, RMI, and EJB.

Originally, SOA involves three actors and three operations. The actors are: service provider, service consumer, and registry, whereas the operations are: publish, find, and bind. SOA enables sharing of capabilities provided as services.

With respect to SaaS, SOA would fulfill the following:

1. Consumers, providers and registry are all services, where Interaction and communication mechanisms are based on messages to exchange documents
2. Composition and service organization and management, including inventory and discoverability mechanisms

However, for SOA to fulfill these requirements, its services need to be shaped with some principles by SO.

2.3 Quality Attributes of SOA-Based SaaS

Satisfying the functional requirements and quality attributes is of paramount importance to the design of SOA-based solutions (Bianco et al., 2007). Hence, the research field of quality attributes of SOA is recently growing. For example, in O'Brien et al. research work (O'Brien et al., 2005), the authors have examined the relationship between SOA and quality attributes. However, within SOSE community, current research on design decisions about the quality attributes is seldom discussed.

In order to design SOA in practice, developers should fully grasp the quality attributes that have significant impact on design and implementation of SOA, because non-functional requirements remain high priority in design process. Availability, modifiability, performance, security, testability, and usability would constitute a set of quality attributes for SOA.

2.3.1 Availability

Typically, availability is related to system failure and its consequences (Bass et al., 2003). In terms of SOA, availability can be analyzed by dividing into two types: in the consumer side, a service failure can lead to an exception of the application of consumer, whereas in the provider side, if a service fails to work, it may affect the provider's finances and reputation (O'Brien et al., 2005). Normally, the two sides should pay attention to the service's availability and they may find the common solutions: service replication and load balancing (O'Brien et al., 2007). Moreover, applications built by consumer should be designed to handle the exceptions, when the service becomes unavailable.

2.3.2 Modifiability

Modifiability concerns with cost of modification (Bass et al., 2003). SOA adopts loose coupling approach to the linkage between service consumers and providers. As mentioned previously, services are autonomous and platform-independent. Thus, it is easy to achieve the modifiability (O'Brien et al., 2007). However, if the interfaces of services need to be modified, the change may lead to big problems, because service interfaces are registered in broker and are used by applications (O'Brien et al., 2005).

2.3.3 Performance

Performance concerns with response time of system when an event (messages, requests or hardware interrupts etc.) occurring (Bass et al., 2003). In SOA, service consumers and service providers are distributed. Thus, the interaction among them increases the response time, because of the time-consuming in network and the overhead of messaging format. In addition, to find the requested services by accessing broker may increase the total time (O'Brien et al., 2007). As a result, to achieve the performance should have great difficulty and should avoid the requirement.

2.3.4 Security

Security is a system's ability that is able to resists attacks when the system is providing services for users (Bass et al., 2003). Security quality attribute is very important to SOA-based system. In O'Brien et al.'s research (O'Brien et al., 2007); they described four principles about security:

- Confidentiality, only authorized subjects can access services
- Authenticity, only trusted author or sender is responsible for the information
- Integrity, the information is not corrupted
- Availability, service is available in a timely manner

2.3.5 Testability

Testability is an ability that enables developers to simplify the establishment of test and to find faults of program (Bass et al., 2003). SOA makes the test become more complex than ever, because SaaS s distributed in network and implementation of service may not be opened (O'Brien et al., 2007). Furthermore, because the services are discovered and invoked dynamically, bug tracking becomes more difficult. Currently, there are no ideal solutions to all the problems of test in SOA, which calls for more research in the area.

2.3.6 Usability

Usability is concerned with how easy to use software for completing a task (Bass et al., 2003). In design process of SOA, designers should provide user-friendly interfaces for applications. In addition they should take the data granularity, usable services and connecting/disconnection process into account (O'Brien et al., 2005).

SERVICE DEVELOPMENT APPROACHES

3.1. Software Engineering Process Model

A software engineering process model is an important criterion to consider when one approaches a SOA-based solution. It concerns with: (i) the service lifecycle, and (ii) the composition lifecycle. A SOSE development lifecycle would consider planning, analysis and design, construction and testing, provisioning, deployment, execution, and monitoring (Papazoglou and van der Heuvel, 2006).

Current service development approaches have been generally categorized into either a top down or bottom up (including reverse engineering). These types of methods have been appreciably extended to green-field and meet-in-the middle respectively (Brittenham, 2001) as the traditional software development method cannot be blindly applied to web services and SOA (Papazoglou and van der Heuvel, 2006). However, all these approaches have in common three invariant elements.

3.2. The Invariant Elements of the Service Development Approaches

All the approaches for service development consider the following elements that agnostic to any approaches:

1. Service contract, i.e. what value does the service provide to its consumers and what are the messages that request and carry out this value; it constitutes the interface of the service
2. Service business logic, i.e. the software agent that implements the service
3. Tools that generate/develop either the business logic (software agent) given a service, or reversely the service contract given business logic

3.2.1. Service contract

The service contract is a description of the service interface that exhibits the semantics of their functional (e.g. capabilities), non-functional (e.g. quality of service), the message descriptions (or schema) and policy requirements. It mainly defines the messages relevant to the service. A software agent that implements its business logic realizes a service contract.

3.2.2. Business logic implemented by a software agent

Software agent implements the service contract in term of logic. The logic may be an existing piece of software or a new one to develop or to generate by using a tool.

3.2.3. Tool

A tool is a development or preferably a generation automated tool that either generates service contract given a software agent as shown in Figures 1.b, 1.c, and 1.d, or generates (or develops/codes/implements) the software agent of a given service contract as shown in Figure 1.a. For instance, Apache SOAP 2.3 and the JAX-RPC (Java API for XML remote procedure call) implementation, which includes the Apache Axis engine, WSDL2Java, and Java2WSDL are examples of tools. WSDL2Java would create a java class that implements a service.

3.3. Types of Service Delivery Approaches

To cope with SO and SOA, SOSE approaches have extended the top-down and bottom-up approaches to two newer approaches called Green-Field and Meet-in-the-Middle (Brittenham, 2001; Chen and He, 2011; Baghdadi, 2013). This extension depends on: (i) whether the contract already exists (e.g. there exists a specification) or the contract is newly designed, and (ii) whether the logic already exists (there exists a piece of code) or to be developed.

To sum up, this work considers four types of methods to develop service:

3.3.1. Top-Down Method

In a top-down method, a service contract is first re-used (e.g. Java interface, C++ abstract class, even a UML interface, or any API) likely using standard such as IDL (Interface Description Language), then a development tool (preferably a CASE tool) is used to assist in designing and generating the implemented logic (code) as shown in Figure 1.a.

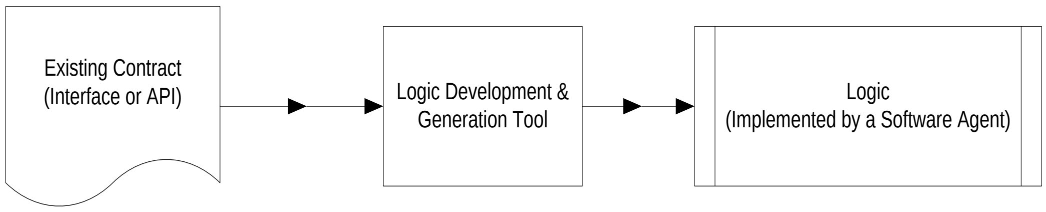
Existing Contract (Interface or API)
Logic Development & Generation Tool
Logic (Implemented by a Software Agent)

Figure 1.a. Process of developing a new WS from an existing interface

3.3.2. Bottom-up method

In the bottom-up method, a new service interface is developed for existing, implemented business logic as shown in Figure 1.b. The business logic can be coded as Java, C++, EJB, or could be a back-end legacy application. For instance, Java classes or EJBs are developed first and then automated tools are used to expose these classes as Web Services (WSs).

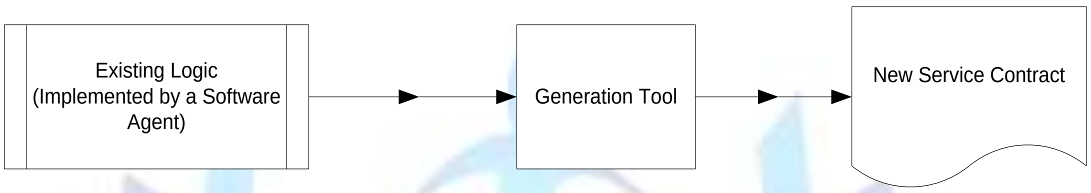

Figure 1.b. Process of generating a service contract from an existing logic

3.3.3. Extended method: Green-field

In the green-field method, a new logic to be provided as service is first designed, implemented, and tested. Then, a generation tool generates a new service contract that will be published as shown in Figure 1.c. Green-field extends the top-down method if there is a need to create a new contract for a new service to be developed from scratch (e.g. new functional requirements, where the solution would be a service).

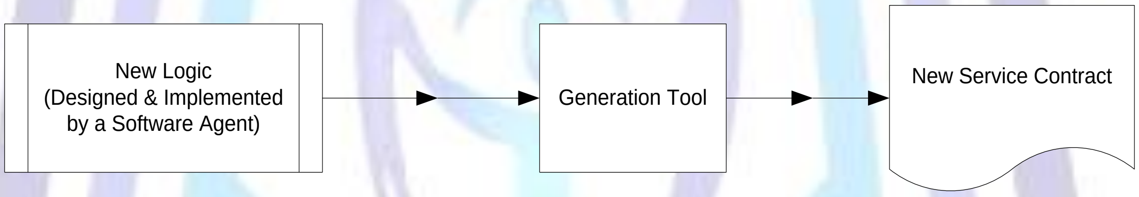

Figure 1.c. Process of developing a new service from scratch

3.3.4. Extended method: Meet-in-the-middle

Meet-in-the-middle is an alternative for bottom-up method, where a service interface already exists and the application that will be used for the service logic already exists. The meet-in-the-middle main task is to map the existing application interfaces to those defined in the service interface definition. This can be done by creating a wrapper for the application that uses the service interface definition, and contains an implementation that maps the service interface into the existing application interface as shown in Figure 1.d. It is a kind of service customization (Chen and He, 2011).

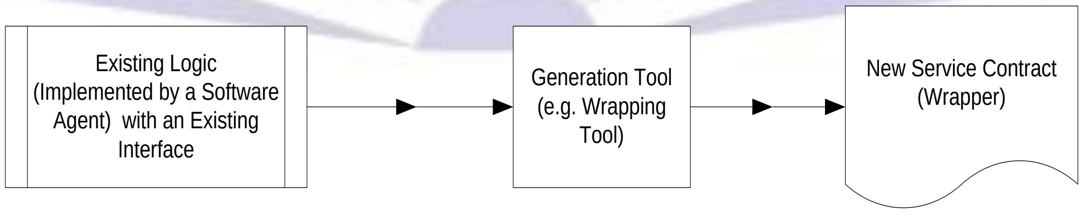
Existing Logic (Implemented by a Software Agent) with an Existing Interface
Generation Tool (e.g. Wrapping Tool)
New Service Contract (Wrapper)

Figure 1.d. Process of wrapping existing logic and interface into WS

3.4. Remarks

A careful method and design decisions is important to ensure that a SOA solution can attain the benefits advertized by its proponents of. The existing types of methods lack some critical perspectives (Baghdadi, 2013). Indeed,

- Business goals and requirements should always drive downstream design, development, and testing. The requirements should be from users rather than design and coding (Baghdadi, 2012b; Chen and He, 2011).
- Though the activities such as build, deploy, and run do exist, a process model that is prone to SOA development is lacking. The existing process models, including Rational Unified Process are no longer

valid for WSs and SOA development. Indeed, SOA is mostly about composition of loosely coupled, interoperable, existing logic, and rarely about creation of new logic.

- None of the types of approaches provides a framework for inspecting a solution; neither have they considered the quality attributes of a solution.

SOADM

This section proposes a new method to achieving quality attributes for designing architecture of services based on design decisions.

4.1. Design Decisions of Architecture

The connection between quality attributes of SOA and architecture of services is not often taken into account (Bianco *et al.*, 2007). Achieving the quality attributes largely depends on the design decisions of architecture. In terms of Bass *et al.*'s description (Bass *et al.*, 2003), they called the design decisions as a set of tactics. Each tactic is applied to achieve some quality attributes. The tactics may be developed from designers' knowledge, skills, experiences or known architectural tactics (Wojcik *et al.*, 2006).

We mainly present the connection between architectural tactics (design decisions) and the requirement for the system quality attributes. We focus on the following issues:

- How do the developers analyze the design decisions based on the quality attributes in a SOA-compliant solution?
- How can these decisions affect the design of service architecture in a SOA-compliant solution?

For instance, we analyze the availability. There are some operations that help to achieve the availability: fault detection, fault recovery and fault prevention (Bass *et al.*, 2003). In order to describe the connection between tactics and quality, we select ping/echo tactic of fault detection as an example. In ping/echo working model, one entity sends a ping to another one and expects to receive the echo. If the echo comes back in time, it can prove that the remote entity and the communication path are both available. Hence based on the ping/echo tactic, we can design two entities in architecture for realizing the fault detection (availability): one is responsible for ping; another is responsible for echo.

In this way, we examine other quality attributes and related tactics. Then, we can know how the tactics influence the design of architecture based on the achievement of quality attributes. In light of the influence, a design method for designing architecture is devised and is named Attribute-Driven Design (ADD) (Wojcik *et al.*, 2006; Bass *et al.*, 2003).

4.2. Classification of Services

A SOA-based system is designed to satisfy functional requirements and non-functional requirement. The service provider is typically responsible for satisfying the functional requirements by offering functional services to user's application. In contrast with service provider, the broker is not directly designed to provide functional services to users, but it is employed to meet the non-functional requirements by offering directory and registry services to users and providers. From this point of view, even though broker only offers non-functional services (directory/registry), it works as a role of a service provider.

Thus, the services in SOA-based system can be classified by functionality as business service and system service.

- Business service is related to those services that are provided by service providers and business service directly takes part in the construction of users' application.
- System service does not directly offer services for users' applications, but it provides the services to entire SOA-based system.

In this way, service broker can be classified as a system service, because it helps to schedule services for applications. In the following content, we present another system services which are designed to satisfy the quality attributes.

4.3. Description of SOADM

SOA-based system consists mainly of all kinds of service (business service and system service). In the system, applications are constructed by managing and composing services which not only include business services but also system services in order to satisfy the both functionalities and quality attributes. According to this objective, we develop a SOA Design Method (SOADM) for this high-level design.

SOADM can be described as a recursive process and is based on attribute-Driven Design ADD method (Bass et *al.*, 2003). The functional requirements and quality attributes constitute the SOADM input; whereas its output is architecture view about services shown in Figure 2. The view shows how many services and what kind of services are needed. In addition, the relationships among the services are defined in the view.

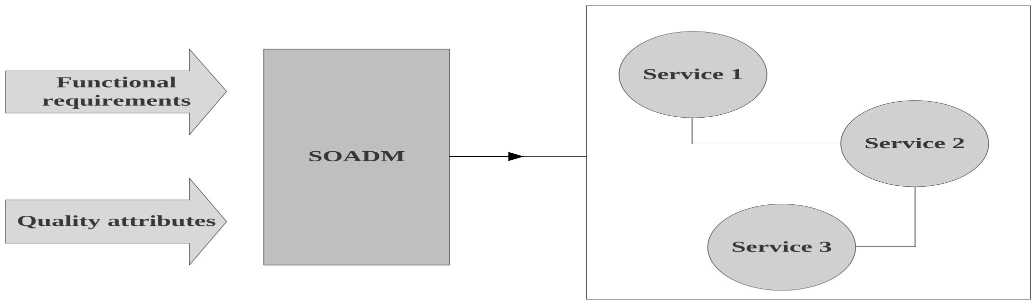

Figure 2. Service-oriented architecture design method

The process steps of SOADM are described below:

Step 1: Collect the requirements including functional requirements and quality attributes that system should satisfy (the collection is made by a specification of the whole system)

Step 2: Choose a requirement from the collection; and delete it from the requirement collection

Step 3: If the chosen requirement belongs to functional requirement then go to step 4, else skip to step 5

Step 4: Add a business service in the architecture view for satisfying the functional requirement; then jump to step 6

Step 5: Find a design decision for the quality requirement, and add proper system services to the view.

Step 6: Verify the architecture view and check if the services in the view need to make composition or decomposition. And then, if it has new requirements, add the requirements to the requirement collection.

Step 7: If the requirement collection is not empty, then go to step 2, else complete the process

SOADM provides architects with a new approach to analyzing and designing a SOA-based system for satisfying both the functionalities and quality attributes in high level design before diving into the implementation of services. In particular, our design method is based on the quality attributes. Thus, it enables the entire SOA system to achieve high quality.

CASE STUDY BASED ON SOADM

In this section, we present a case study that consists of a simple online bookstore application for demonstrating our design method: SOADM. This method handles all the kinds services.

At first, after analyzing the requirements of this application, we assume that we get the requirement collection described as below:

- Client can browser the information of books
- The availability of information of books
- Client can order a book by offering personal information
- The security of payment

We choose the first requirement and input it to SOADM. Then we get a service named as Browse-Catalog shown in Figure 3. Then the requirement collection is updated after the first turn:

- The availability of information of books
- Client can order a book by offering personal information
- The security of payment.

Browse-Catalog

Figure 3. The view of service in first turn

Choose the first requirement and input it to SOADM. Because it belongs to quality attribute, we apply ping/echo tactic (Bass *et al.*, 2003) to satisfy the availability by making sure that the Browse-Catalog service is available. According to the step 6, the echo service merges with Browse-Catalog service. The view of this turn is shown in Figure 4. Then the requirement collection is updated after the second turn:

- Client can order a book by offering personal information
- The security of payment

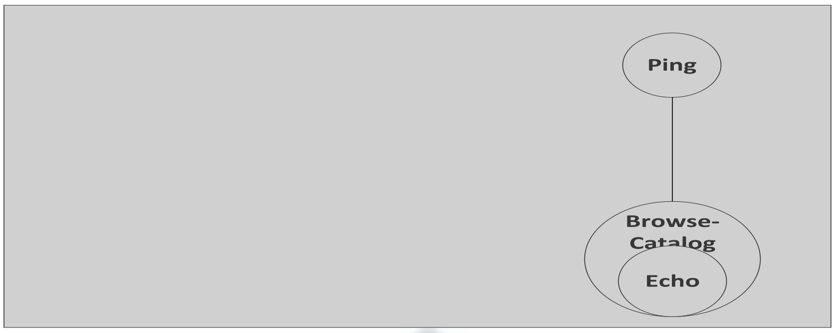

Figure 4. The view of service in second turn

Like the previous turns, we get a new service named Order-Service shown in Figure 5. Then the requirement collection is updated after the third turn:

- The security of payment

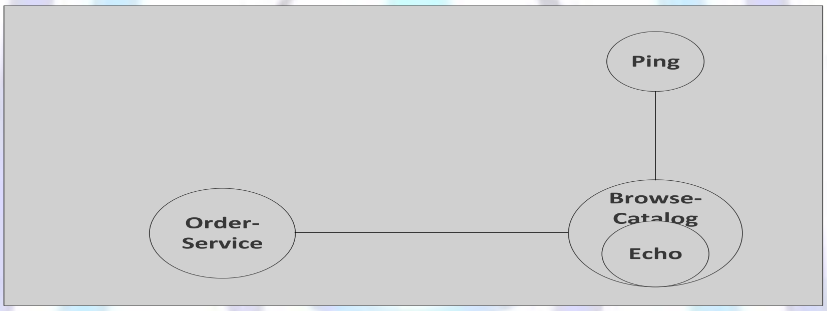

Figure 5. The view of service in third turn

To deal with the last requirement, we input it to SOADM. In this turn, we add a security service to meet the quality attribute. The entire view of services is shown in Figure 6. Then we finished the design process.

Browse- Catalog

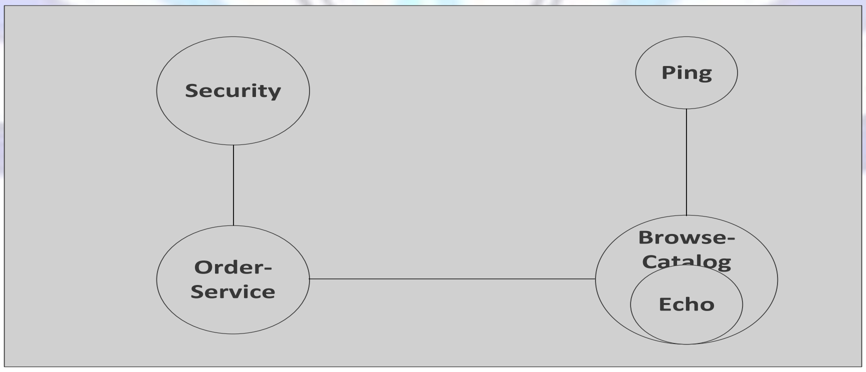

Figure 6. The view of service in last turn

RELATED WORK

In early stage of SO, many authors focused on a developing process of Web services following by extending traditional approaches (Arsanjani *et al.*, 2008; Papzoglou and van der Heuvel, 2006; Erl, 2008; Wirsing *et al.*, 2008).

Few comprehensive surveys on WS development approaches have been realized. Recently, Gu and Lago in (Gu and Lago, 2011) presented a framework for comparing the existing SOA methods in order to specifically highlight aspects that are specific to SOA They aimed to differentiate the methods that are truly service-oriented from those that deal little with service aspects. As such, they suggest using the criteria defined in the framework as a checklist for selecting a SOSE methodology. The same authors in (Gu and Lago, 2010) proposed a literature review on service identification. The classification and comparison of 30 identification methods showed significant heterogeneity in the input/process/output of the studied methods. They reported in their conclusion the necessity of future research directions to improve the existing methods. Kohlborn *et al.* in (Kohlborn *et al.*, 2009) have reviewed 30 service analysis approaches to conclude that a comprehensive approach to the identification and analysis of both business and supporting software service is missing. They proposed a consolidated approach to business and software services and identification that combine the strengths of the examined approaches.

In (Al-Rawahi and Baghdadi, 2005), the authors have distinguished business-oriented approaches from technology-oriented approaches.

In (Papazoglou and van der Heuvel, 2006), the authors provide an overview of the methods and techniques used in service-oriented design and development in order to provide a methodology from the point of view of both service consumer and service provider.

In (Brittehham, 2001), the author describes an approach for developing services from the point of view of the service providers and service requestors. The development approach explains the components, interactions, application development patterns and tools necessary to implement WSs in general. In this work, the approaches have been categorized into green-field, top-down, bottom-up and meet-in-the-middle depending on the existence or not of the service interface and the service implementation. Later Baghdadi (2013) compared these four approaches.

The proposed framework aims at providing a set of sound SOSE perspectives that any methods need to consider. It is meant to reuse and extend the main strengths of the different approaches and tools to guide a method geared towards producing SaaS.

The framework sketches out the elements of a SOSE method, and finally proposes SOADM, a method for developing services that comply with SO and SOA.

CONCLUSION

This work has first compared four types of service development methods within a framework that consists of a set of critical elements that any SOA-based system development method should consider in order to provide a solution that conforms to SO. The framework has showed that the existing types of methods lack some critical perspectives. A process model that is probe to SOA development is lacking. Moreover, none of the types of methods consider different views of a solution, neither have they provided a framework for inspecting a solution by assessing the quality attributes of a solution.

This work has discussed the issues of quality attributes and design decision in SOA-based system. We mainly concentrate on the problems: how to analyze the design decisions based on quality attributes and how design decisions impact on the service in SOA-based system. In order to resolve the problems we developed a method to design of SOA-based system for achieving both functionalities and quality attributes in high level design. The design approach can provide architects with high level view about desired services that will be designed and implemented in detailed design. At last, the paper presents a case study about a simple online bookstore by using our design approach. After adopting the recursive process, we get a clear service view for fulfilling the functional and quality requirements.

In our future work, we are going to improve our design approach and apply it to a complicated SOA-based system for evaluating our research.

In addition, this work would be further developed in exploring and extending the aforementioned open issues towards a service science that considers the service as a reusable value in order to provide a comprehensive methodology for developing business or technology solutions as service systems with respect to SO and SOA.

REFERENCES

[1] Al-Rawahi, N. and Baghdadi, Y. 2005. Approaches to Identify and Develop Web Services as Instance of SOA Architecture. In Proceedings of the IEEE-ICSSSM05, Vol. 1: pp 579-584, Beijing, China, June 13-16, 2005.

[2] Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Gariapathy, S., and Holley, K. 2008. "SOMA: a method for developing service-oriented solutions", IBM Systems Journal, 47(3), pp.377–396.

[3] Baghdadi, Y. 2014. "Service-Oriented Software Engineering: A Guidance Framework for Service Engineering Methods", Int. Journal of Systems and Service-Oriented Engineering. In Press.

[4] Baghdadi, Y. 2013. "A comparison framework for service-oriented software engineering approaches: issues and solutions", Int. Journal of Web Information Systems, 9(4), 279-316.

[5] Baghdadi, Y. 2012a. "A survey on approaches to identify and develop web–enabled services with respect to service–orientation paradigm and SOA: towards a value–oriented approach", Int. Journal of Computer Applications in Technology 45.1 (2012): 1-14.

[6] Baghdadi, Y. 2012b. "A Methodology for Web services-based SOA realization", Int. Journal of Business Information Systems, 10(3), 264-297.

[7] Baker, J., Jones, D., Cao, Q., and Song, J. 2010. "Conceptualizing the Dynamic Strategic Alignment Competency", Journal of the Association for Information Systems, 12(4).

[8] Bass, L., Clements, P. and Kazman, R. 2003. Software Architecture in Practice. Second Edition, Addison-Wesley.

[9] Bell, M. 2008. Service-Oriented Modeling: Service analysis design and architecture. Wiley.

[10] Bianco, P., Kotermanski, R., and Merson, P. 2007. Evaluating a Service-Oriented Architecture', Technical Report CMU/SEI-2007-TR-015.

[11] Brittenham, P. 2001. Web services development concepts. IBM Software Group.

[12] Chen, H. and He, K. 2011. "A Method for service-Oriented Personalized Requirements Analysis", Journal of Software Engineering and Applications, Vol. 1, pp. 59-68.

[13] Cohen, S. 2007. "Ontology and Taxonomy of Services in a Service-Oriented Architecture", The Architecture Journal, vol. 11, pp. 30-35. Microsoft press, 2007.

[14] Erl, T. 2008. Web Service Contract Design and Versioning for SOA. Prentice Hall.

[15] Gu, Q. and Lago, P. 2010. Service identification methods: A systematic literature review, in Service-Wave 2010: Towards a Service-Based Internet, ser. Lecture Notes in Computer Science, vol. 6481. Springer Berlin / Heidelberg, pp. 37–50, 2010.

[16] Gu, Q., & Lago, P. 2011. Guiding the selection of service-oriented software engineering methodologies. Service Oriented Computing and Applications, 5(4), 203-223.

[17] Kohlborn, T., Korthaus, A., Chan, T. and Rosemann, M. 2009. Identification and analysis of business and software services—a consolidated approach. IEEE Transactions on Services Computing 2(1), 50–64.

[18] Marks, E. and Bell, M. 2006. Service Oriented Architecture: A planning and implementation guide for Business and Technology. John Wiley & Sons.

[19] O'Brien, L. Bass, L. and Merson, P. 2005. Quality attributes and service-oriented architectures, Carnegie Mellon University, Software Engineering Institute.

[20] O'Brien, L., Merson P. and Bass, L. 2007. Quality Attributes for Service-Oriented Architectures. In Proceedings of SDSOA '07 Proceedings of the International Workshop on Systems Development in SOA Environments, IEEE.

[21] OASIS. 2006. A reference model for service oriented architecture', Billerica, MA: Organization for the Advancement of Structured Information Standards.

[22] Papazoglou, M. P., Andrikopoulos, V. and Benbernou, B. 2011. Managing Evolving Services. IEEE Software 28(3), pp. 49-55.

[23] Papazoglou, M. P. and van der Heuvel, W. J. 2006. "Service-Oriented Design and Development Methodology", Int. Journal of Web Engineering and Technology, 2(4), 412-442.

[24] Patterson, D. and Fox, A. 2012. Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing. Strawberry Canyon LLC, UC Berkely.

[25] Wojcik, R. Bachmann, F. Bass, L., Clements, P., Merson, P., Wood, R. Attribute-Driven Design (ADD), Version 2.0. TECHNICAL REPORT,CMU/SEI-2006-TR-023, ESC-TR-2006-023, 2006.

[26] W3C. 2010. Web services architecture usage scenarios'. W3C Working Group Note. 11 February 2004. Available at: http://www.w3.org/TR/wsarch-scenarios.

[27] Wirsing, M., Hoelzl, M., Koch, N., Mayer, P. & Schroeder, A. 2008. Service engineering: the sensoria model driven approach. in: Proceedings of Software Engineering Research, Management and Applications, Prague, Czech Republic, pp. XIV–XVI.

Author' biography with Photo

Dr. Hamid Mcheick is working as an associate professor in Computer Science department at the University of Québec at Chicoutimi, Canada. He has more than 18 years of experience in both academic and industrial area. He has done his PhD in Software Engineering and Distributed System in the University of Montreal, Canada. He has a master degree in computer science from University of Quebec At Montreal (UQAM), Canada and a master degree in mathematics and computer science from Lebanese University, Lebanon. He is currently working on Pervasive computing, Service-oriented computing, Web Applications, Cloud Computing, Natural Disaster, Software architecture, Software development methodologies, Software evolution and maintenance and Aspect-oriented software engineering. He has supervised many post-doctorate, PhD and master students. He has six book chapters and more than 26 research papers in international journals and 65 research papers in international/national conference proceedings in his credit. Dr. Mcheick has given many keynote speeches in his research area, particularly in Distributed Middleware, Software Architecture, Software Connectors, Service Oriented Computing and Cloud Computing. For example, he has given many talks in Canada, China, Italy, and Tunisia. He has also received many awards and congratulations of his quality of research (UQAC, ICCIT-IEEE2012, ICCIT-IEEE2011, University of Jordan, FASE2006), and quality of teaching (UQAM, UQO). He is a chief in editor, chair, co-chair, reviewer, and member in many organizations (IEEE, ACM, Springer, Inderscience), journals (IJCNDS, CSENG, IJCSI, JTAER, IJWS, etc.) and conferences (CCECE-IEEE, ICISA-Springer, ATNAC-IEEE, ACC-ACM, ICCIT-IEEE, etc.) around the world.

Dr. Youcef Baghdadi received his PhD degree in computer science from University of Toulouse 1, France. His is currently an Associate Professor with the Department of Computer Science at Sultan Qaboos University in Oman. He published many articles in journals such as the Information Systems Frontiers, Information Systems and E-Business, Service-Oriented Computing and Applications, Business Information Systems, Electronic Research and Applications, Web Grids and Services, and others. His research aims at bridging the gap between business and information technology (IT), namely in the areas of cooperative information systems (IS), web technologies, e-business, service-oriented computing, and methods for service oriented software engineering.