Computer and Information Science; Vol. 7, No. 1; 2014
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

A Multifaceted Web Services Architecture: Toward a Meta-Service Framework for Service and Composition Development

Youcef Baghdadi[1]

[1] Department of Computer Science, Sultan Qaboos University, Muscat, Oman

Correspondence: Youcef Baghdadi, Department of Computer Science, P.O. Box, 36 – PC 123, Al-Khod, Oman. E-mail: ybaghdadi@squ.edu.om

Received: December 4, 2013 Accepted: January 5, 2014 Online Published: January 22, 2014

doi:10.5539/cis.v7n1p172 URL: http://dx.doi.org/10.5539/cis.v7n1p172

Abstract

SOA is an architectural style that promotes solutions, especially service-based applications to support flexible business processes, where services are loosely coupled and interoperable components. Many methods have been developed in industry and academia. Yet the comparison frameworks show that these methods neither comply with SOA nor service orientation principles. Therefore, we need theoretical frameworks to guide developing methods and processes. A meta-service framework, representing the relationships between all the elements of a multifaceted services stack from different perspectives, would guide a development process. We propose an architecture made up of four interrelated facets that are: (F1) service contract description, (F2) modernization of legacy applications and databases, (F3) deployment and management, and (F4) service based applications by composition, including orchestration and choreography. Each facet is considered as a level of abstraction that is concreted into a phase of the process.

Keywords: web services, service-orientation, service oriented architecture, service oriented computing, service-based applications, composition, development process

1. Introduction

Web services technologies use the Internet and standard XML-based languages such as WSDL, BPEL, or WS-CDL, to constitute the basic components and a suitable realization of Service Oriented Architecture (Erl, 2008; Baghdadi, 2012b). SOA constitutes a response to the growing needs for greater reuse, integration, and composition of IT services to support business flexibility and agility (Cummins, 2010; Welke et al., 2011). To conform to SOA, Web services need to be loosely coupled and interoperable. Service Science (SS) shapes the concept service with fundamental, generic properties (Spohrer et al., 2008). Service Orientation (SO) provides a set of service design principles (Erl, 2008; Papazoglou et al., 2008). These properties and principles enforce loose coupling and interoperability of services, in order to allow services to be the basic components of Service-Based Applications (SBAs), built with respect to an architectural style that is Service-Oriented Architecture (SOA).

Yet, developing SBAs requires an engineering approach; referred to as Service-Oriented Software Engineering (SOSE). It mainly concerns with developing methods that encompass the main aspects of SOSE that are: (i) the engineering of the services as basic components, (ii) the engineering of compositions of services as SBAs (Papazoglou et al., 2008), (iii) the management of services and SBAs (Papazoglou et al., 2011), and (iv) the quality of both services and compositions (Bianco et al., 2007).

As expected, many methods in both academia and industry have been developed. Most of the well-known methods such as SDLC (Papazoglou & van den Heuvel, 2006), Sensoria (Wirsing et al., 2008), SOAF (Erradi et al., 2006), SOMA (Arsanjani et al., 2008), Erl's methodology (Erl, 2009), claim their compliance with SOSE, namely a development process that produces services and SBAs that comply with SO and SOA. Yet, comparison frameworks such as those developed by Baghdadi (2012a) and Gu & Lago (2011) show that the output of these methods, when applied, does not fully comply with SO and SOA. Indeed:

- They do not explicitly reflect each role in SOA, the service provider, the service consumer, and the service registry. Their respective process concentrates on the provider of the service, while the consumers and the registry not only have a role but also constrain SOA.

- They vary significantly in terms of their coverage to SOSE aspects. Some methods cover only services as components, whereas others cover secrecies and SBAs.
- They vary in their delivery strategies, i.e., the generic process to develop services such as meet-in-the-middle, top-down, bottom-up, or green-field (Baghdadi, 2012a; Brittenham, 2001).

The existing methods have been engineered using approaches, such as consolidation (Kohlborn et al., 2009), or extension (Sensoria (Wirsing et al., 2008) or SOMA (Arsanjani et al., 2008)), product line (Mazo et al., 2012). Although, these engineering approaches are proven useful for engineering methods based on function, object, or components as the main building blocks, they are pertinent to only limited perspectives and aspects of SO. These methods are limited to certain perspectives of services, namely their identification and construction (Gu & Lago, 2011; Al-Rawahi & Baghdadi, 2005). This is mainly owing to a lack of a framework (abstract model) that captures the relevant elements of a SOSE environment, the relationships between the elements, and the constraints (if any) into what we refer to as Meta-Service Framework (MSF).

In this work, we propose a multifaceted Web Services Architecture (WSA) towards a MSF that captures a sound set of Web service properties. From specification to reuse into SBAs through wrapping and deployment, having in mind that SOA is in big part for reusing and integrating existing enterprise assets such as legacy applications and databases.

The MSF sketches out the service properties from different facets, where a facet represents different abstraction, benefits and interests. These are (F1) service contract, (F2) service as IT-component, (F3) services representing legacy systems, and (F4) SBAs that support business processes (BPs).

The MSF aims at guiding service and SBA development, whereby each facet is considered as a phase in the development process.

The remainder of the paper is as follows: section 2 describes the related concepts to SOSE. Section 3 details the proposed MSF. Section 4 shows how the MSF guides the phases of the process through a multi-level architecture. Section 5 presents related work. Finally, a conclusion section presents some open issues and further development.

2. Service Oriented Software Engineering Concepts

This section elaborates on the concepts and paradigms related to SOSE, as relevant entities in the MSF that would guide in developing Web services and SBAs. These are Service Science (SS), Web services, (WSs), SO, SOA, and Service Oriented Computing (SOC). Therefore, we need a deep understanding of these entities, especially their relationships with each other in order to sketch out how software system solutions are built by composing services, and consequently the process to build those solutions.

2.1 Web Services

From (W3C, 2010), we can compile a set of properties to define a Web service as a computational resource that is identifiable by a unique identifier. It performs one or more tasks through an agent (e.g., a provider agent that is a software unit); and is used by a requester agent. It has a description and an interface. The description is a machine-processable description of the service interface that shows the messages that are exchanged by the service. The interface defines the messages relevant to the service, that is, the different types of messages that the service sends and receives, along with the message exchange patterns that may be used. In addition, a service description may include one or more policies applied to it. Finally, a description is expressed in WSDL. In addition, services should: (i) define, in a machine-readable standard, the semantics of their functional (e.g., capabilities), non-functional (e.g., quality of service), and policy requirements, making the service contracts understandable by the consumers and governed by policies to which the consumers adhere, and (ii) communicate through SOPA, messaging protocol built on top of the Internet protocols.

2.2 Service Science (SS)

SS is expected to providing theories and methods from many different disciplines to study the service as phenomenon (Spohrer et al., 2008). We build on the state of art, where the concept service is already in use, namely in business and IT, to capture: (i) some service intrinsic, core conceptual properties, including at least its structure, behavior and semantics, and (ii) usages, utilities, and in-sourcing. This would allow a definition and accordingly an easy way for identification of contracts that should be understood by the consumers and governed by policies to which the consumers adhere.

2.3 Service Orientation

Erl (2009) defines SO 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 a service-oriented solution. To comply with SO and to be an element of composition, a service should exhibit some desirable principles in addition to its fundamental properties, which enforces: loose coupling, interoperability, and separation of concerns as detailed in Table 1.

Table 1. How the desirable service principles promote the SO principles

Service Principles	SOA Principles		
	Loose coupling	Interoperability	Separation of concerns
Standard Contract	√		
Sharing and Reuse	√	√	√
Discoverability	√	√	
Abstraction	√	√	√
Autonomy	√	√	√
Stateless	√		
Messaging	√	√	
Self-contained	√	√	√
Granularity	√	√	√

These principles enable SOA, which itself promotes solutions as compositions of principled services.

2.4 Service Oriented Architecture (SOA)

SOA enables sharing of capabilities provided as services. With respect to SOSE framework, SOA would fulfill the following:

(1) Consumers, providers, and registry are services.

(2) Interaction and communication mechanisms are based on messages to facilitate operations such as publish, find, match, and bind for efficient interoperability.

(3) Service management artifacts, including services inventory (e.g., adapted registry).

(4) SBAs and service organization and management, including inventory and discoverability mechanisms.

To adapt SOA as a computing architecture that applies to SOSE, the aforementioned principles, originally intended to provide guidance for the design and development of services, need to be realizable through a distributed computing paradigm. This is the role of SOC.

2.5 Service Oriented Computing

In the proposed framework, SOC (a distributed computing platform) is expected to examine and make services computational components that can participate in different composition to develop SOSs (Papazoglou et al., 2008). That is, making existing units of software, including legacy applications, transparent with respect to their location and technology platforms (e.g., networks, operating systems and programming languages) in order to be provided as services, whereby each service participates in different compositions of software and in different contexts.

Therefore, SOC will provide structural and behavioral elements such as service programming model, service bus, service interactions, selection, discovery, composition, management, organization, security, and exception handling, which we categorize into: (1) a set of horizontal services that are application-independent, and (2) a set of platform services that allow services to communicate.

2.5.1 Horizontal Services

All Web services use horizontal services independently of any SOS solutions. These are:

HS1 Service management: SOC provides mechanisms for management and monitoring to enable mastering the evolution of services (Baghdadi, 2007; Papazoglou et al., 2011). Without such mechanisms, services performance and stability will degrade, and accordingly software systems composed out of them.

HS2 Service organization: SOC should provide artifacts to organize and manage services for a very large internal and external reuse. The object-orientation paradigm of organizing classes into hierarchy is no longer adequate (Arsanjani al., 2008).

HS3 Service security management through WS*.

HS4 Service concurrency management.

HS5 Composition and composition management, including coordination, choreography, or orchestration.

2.5.2 Platform Services

The platform services allow deployed services to communicate within a composition. These are:

PS1 Interface definition

PS2 Service communication

PS3 Exception handing management

PS4 Service selection

PS5 Service discovery

PS6 Service addressing

PS7 Execution, execution describes how the realization of a service is carried out

These properties facilitate the SBAs as compositions of Web services.

2.6 Service-Based Applications (SBAs)

Web services are developed to be reused in composing more coarse-grained web services or SBAs to support BPs, whereby a BP is a set of coordinated activities. Each composition has a style that may be an orchestration or choreography. An orchestration refers to an executable BP. It describes how web services invoke each other, under the control of one service. In the orchestration style, the BP is controlled and executed by one of the business parties involved in the process. Choreography is associated with the message exchanges between multiple BPs. In web services technology, Business Process Execution Language (BPEL) represents orchestration and Web Service Choreography Description Language (WS-CDL) represents choreography. A BPEL is executable, whereas WS-CDL is just a description of collaboration between partners.

However, SBAs should be developed in compliance with an architectural style that is SOA, where both consumer and provider have a role to play (Baghdadi, 2012a) as shown in Figure 1.

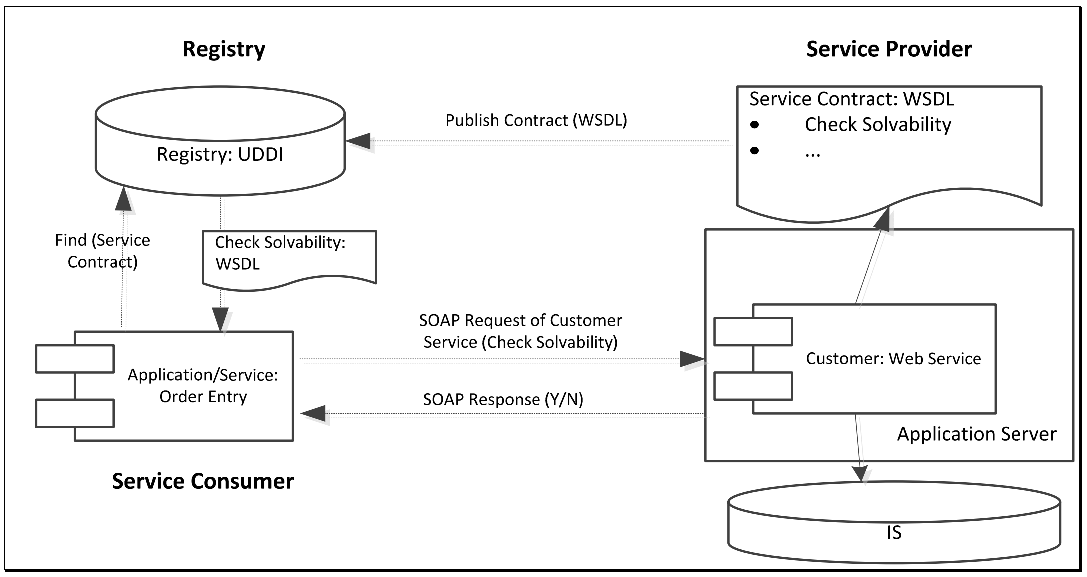

Figure 1. SOA instantiated with web service: ‘Customer’ is exposed as a service

Therefore, services principled by SO would be a realization of software systems if we provide techniques to:

(1) Develop services mainly by wrapping candidate functionalities (Baghdadi & Al-Bulushi, 2013). Generally these functionalities exist in the legacy systems, however if some functionalities do not exist, we need to develop them as services. The services are then deployed and registered in a private/public registry.

(2) Check the compliance of the wrapped Web services with the SO principles as shown in Table 1.

(3) Develop SBAs to support BPs by composing the developed, deployed and registered services that are principled by SO and with respect to SOA. The standard BPEL may be used to describe and execute SBAs, whereas the standard WS-CDL may be used to describe a crossing BP.

The aforementioned concepts and paradigms have helped in shaping services, the architecture of solutions built by reusing services; and how these services and solutions are deployed and executed in a distributed platform that is SOC platform.

Next, we need to provide a meta-model that captures all the properties of all these entities into multifaceted architecture; we referred to as MSF that guides a step-wise process for developing both services and SBAs.

3. Meta-Service Framework for Services and SBA Development

The MSF described in this work as shown in Figure 2 is built on the work described in (Baghdadi, 2009). It represents the Web services as pieces of software generated by wrapping existing systems (applications or databases), the specification of the contract: both abstract and concrete parts, and their deployment into their respective servers and registries to be further reused in developing SBAs. For a MSF to play the role of a framework that guides the development of SBAs based on modernizing existing systems, notably by wrapping, and composition techniques, it needs to capture the properties of the Web services from different facets, namely:

(F1) Service contract (Purple surrounded in Figure 2)

(F2) Service as IT-component (Red surrounded in Figure 2)

(F3) Services representing legacy systems to be modernized (Green surrounded in Figure 2)

(F4) SBAS that support business processes (BPs) (Blue surrounded in Figure 2)

3.1 Service Contract Facet

Service contract is a machine-processable description of the service interface description. It exhibits the semantics of its functional (e.g., capabilities), non-functional (e.g., quality of service), the message description (or schema), and policy and agreement requirements. We distinguish the abstract part (e.g., what is the purpose of the service and its capabilities) from the concrete part (e.g., how and where can the service be accessed). This distinction is very important, as an abstract part may be: (i) realized differently (e.g., location and protocol), (ii) designed from scratch (by the provider), (iii) reused from and existing industry standard, or existing package (e.g., IDL/ java/UML interface, abstract class, Oracle package), or (iv) developed/generated and deployed separately from the concrete part. For instance, many services may implement the abstract part of the interface (describing one operation, the message in and the message out), each service has its own location (where) and access and communication protocol (how). In addition, the same abstract part could exist as a local java/UML interface or as an industry standard interface.

3.2 Deployment Facet

Web services are deployed within a run-time server. It could be a standalone or within an application server. A server consists of a Web service container that manages the life cycle of the application implementing the services, and a SOAP processor that processes the exchanged messages as distinguished in Figure 2.

The Web services container is responsible for:

- Managing the life cycle of the application implementing the service
- Generating the WSDL which will be registered in the UDDI, where the client applications can find it and generate a client proxy. At run time, the client uses the proxy to construct and send SOAP message to the Web services

The SOAP processor is responsible for:

- Processing of incoming messages
- Converting from XML into native PL data types
- Routing the request to the application that implements the service

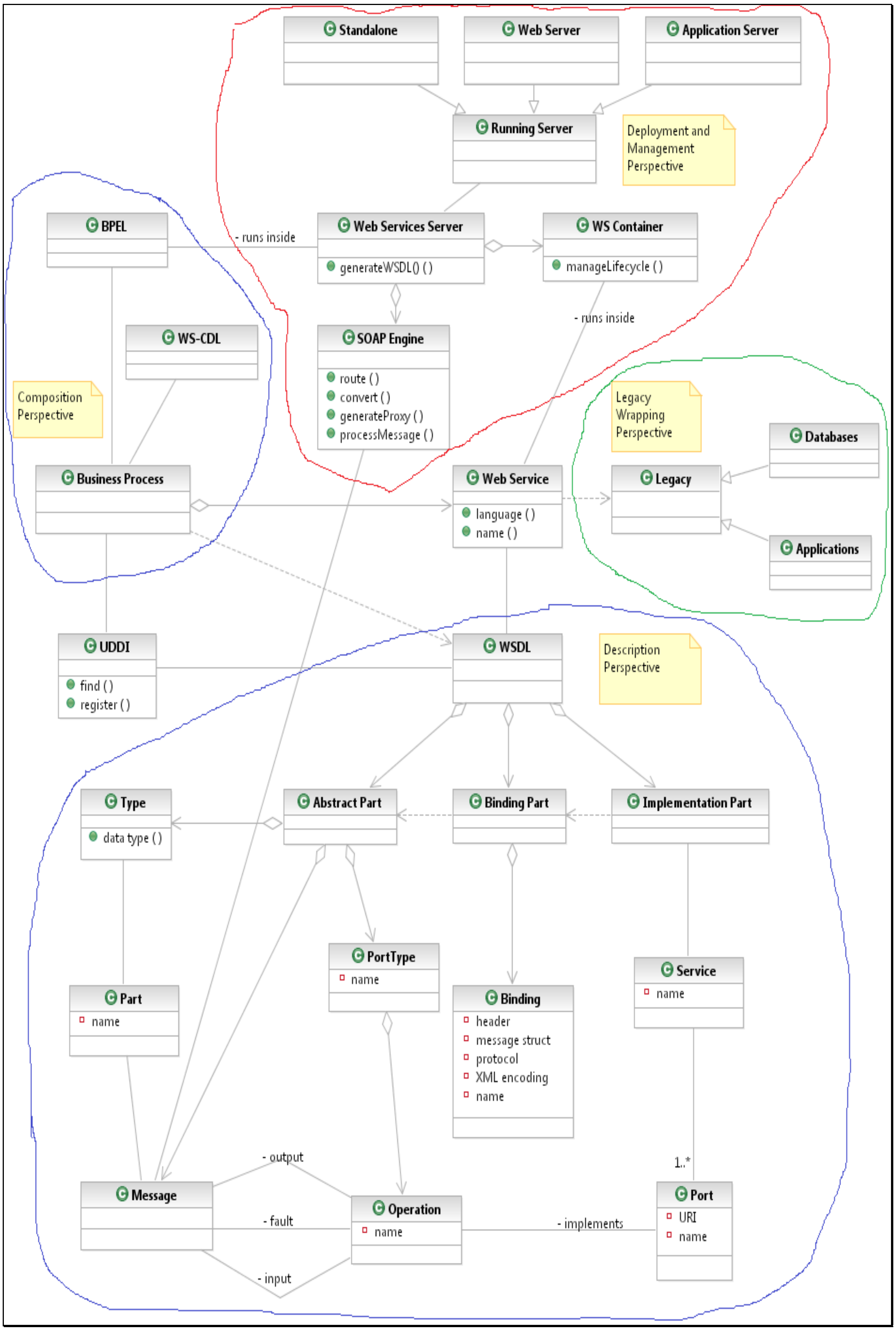

Figure 2. A UML Class diagram representing the four facets of MSF (each facet has a distinct color)

3.3 Modernization of Legacy System Facet

The term 'legacy application' describes old applications and databases, running on an outdated platform and infrastructure that continue to be used because of the costs and risks of replacing or redeveloping them are high. For instance, applications that are meeting the organizations requirements, preserving IT investment, reliable, and secure, yet could not be reusable with SOA. Thus, moving to SOA with web services would not be possible without taking into account such legacy applications and databases, for several reasons, namely: (1) they perform, (2) they embed most of the business functions, and (3) they are built at high cost; and we need to preserve investments on them. The UML diagram in Figure 2 distinguishes the Web services wrapping perspective.

3.4 SBA as a Composition Facet

Reusability is one of the major properties of Web services since services can be reused instead of being rewritten in the Web services environment. Therefore WSA can be viewed from a usage point of view to provide a flexible software composition implementing BPs. The UML diagram of Figure 2 distinguishes the Web services composition perspective, where software systems supporting BPs is presented, with respect to SOA, with BPEL (for orchestration) or WS-CDL (for choreography) of Web services registered in a registry (e.g., UDDI). The consumer and provider of Web services use SOAP to communicate with each other. Indeed, Web services can be combined in two ways: orchestration or choreography

3.4.1 Orchestration

In orchestration, a central coordinator represents the BP. This coordinator is also a Web service that coordinates and controls the execution of the required operations from the other Web services involved (aka partners) in the composition (BP). The specification of the coordination and communication are within the coordinator Web service, i.e., the unique Web service that has all the information about the composition and the flow of execution of the operations provided by the other Web services. That is the involved Web services are not aware of the composition and know nothing about it.

A Business Process Execution Language (BPEL) process specifies the involved Web services and the order in which partner Web services are invoked. It also specifies how the partners are invoked, i.e., in sequence or in parallel.

BPEL is an executable language having its own variables, flow control statements, invocation statements. It is similar to BPMN and UML activity diagram.

For a composition, a BPEL Web service:

- Receives a request from the trigger of the BP (itself a Web service)
- Processes it by invocation of different partners in sequence or in parallel
- Responds to the trigger
- BPEL relies on the WSDL description of the Web services invoked as shown in Figure 2.

3.4.2 Choreography

Unlike, orchestration, choreography does not have a coordinator. That is, each of the Web services involved in the choreography knows when to execute the required operations and with which Web services it interacts. Choreography is rather a description of collaboration between Web services, where it focusses on the exchange of messages in cross-boundary BPs. All Web services that participate in the choreography need to be aware of the BP, the operations to execute, the messages to exchange, and when the messages need to be exchanged.

From the perspective of composing Web services to execute BPs, orchestration is a more flexible paradigm and has the following advantages over choreography:

- A coordinator manages the coordination of the Web services that compose the solution supporting the BP.
- Partner Web services can be included in the composition without any knowledge about the composition they are serving.
- Partner Web services can be replaced in case of non-reliability or availability of one of the Web services involved in the composition, i.e. in case of expectations.

These four perspectives represented in the meta-model for WSA, shown in Figure 2, provides guidance towards SOS development mainly based on three techniques:

(1) Wrapping of legacy systems into Web services

(2) Checking the compliance of the wrapped Web service with the SOC principles

(3) Composing service-oriented software out of the Web services

4. Guidance to Develop SBAs

The above-mentioned MSF provides guidance for processes to develop both services and SBAs toward the application of a SOA maturity model, at least the initial level (Welke et al., 2011). The activities of the process are: specification, design, implementation and test, deploy, run, and manage. It is worth noting that a web service can be developed as: (i) a new web service from scratch, (ii) by modernizing existing application, or (iii) by composing a new service out of the existing service. Similarly, a service contract may be developed from scratch (new service contract), or from an existing local or standard interface. Thus, there exist several delivery approaches as detailed in the next section.

4.1 Delivery Approaches for SBAs

With respect to SOA and web service, the delivery approach, i.e., the engineering strategy depends on: (i) whether the service exists or not, i.e., whether the business logic that implements the service (e.g., class/component, or legacy application, or any piece of code) exists or not, and (ii) whether the service contract exists or not, i.e., whether the specification of the abstract part exists (either locally or somewhere as standard interface) or not. When we combine these alternatives, we end up with four approaches (engineering strategies) to first develop services (Baghdadi, 2012a; Brittenham, 2001; Papazoglou & van den Heuvel, 2006), then composition of SBAs. These are Top-down, Bottom-up, Green-field, and Meet-in-the-middle approaches. The Green-field extends the Top-down approach, whereas the Meet-in-middle extends the Bottom-up.

4.1.1 Top-down Approach: Service Contract Exists and the Web Service Is New (to Develop)

In a top-down approach, we need to develop new service from an existing contract (abstract part). A service contract (e.g., Java interface, or an industry standard interface) is first located. Next, it is re-used by creating a template of it. Then, based on the template, a development tool implements the service that is finally deployed within a server, whereby the concrete part of the contract is generated and published.

4.1.2 Bottom-up Approach: Service Exists and the Service Contract Is New (to Develop)

In the Bottom-up approach, a new service contract is developed and published for an existing service. It represents implemented business logic (e.g., legacy application). First, the business logic is located. Then, the service contract is generated by using a tool. This results in exposing the legacy applications as web services. This may require guidance for wrapping techniques such the ones developed in (Baghdadi & Al-Bulushi, 2013; Lewis et al., 2008).

4.1.3 Green-Field: Both Service and Service Contract Are New

The Green-field approach extends the top-down approach if there is a need to create a new web service contract for new service to be developed from scratch (e.g., new functional requirements, where the solution would be a service). The web service is first developed. Then, the service provider generates the web service contract (abstract and concrete part) from the deployed web service. This type of web service contract is created from scratch.

4.1.4 Meet-in-the-Middle: Both Service and Service Contract Exist

The Meet-in-the-middle extends the Bottom-up approach, where both the service contract and the application that will be used for the web service exist. The main task is to transform the existing application interface to that defined in the web service contract.

4.2 The SBA Process Guided

In addition to presenting and highlighting the multiple facets of Web services in a comprehensive architecture, the MSF is a kind of framework that guides service engineering development processes, including service development process and SBA process as shown in Table 2.

(1) The contract facet helps in describing the service from both business and IT perspective by using a green-field approach or a top-down approach to identify the service in term of contract (abstract part) then use Web services underlying technologies to construct it.

(2) The modernization facet indicates the guidance process for using a modernization technique among many existing ones (Baghdadi & Al-Bulushi, 2013; Canfora et al., 2008; Comella-Dorda et al., 2000; Rahgozar & Oroumchian, 2003; Lewis et al., 2008). This guidance would first locate the business functions locked within

legacy applications to modernize them. The contact could have been already specified in the previous step. Then, use tools to extract and wrap the business functions. It is then a meet-in-the middle approach.

(3) The deployment facet guide toward the required platforms and infrastructures used to deploy and run services then test them. It also guides the management and monitoring of the deployed services.

(4) The SBAs facet describes the different scenarios in composing the applications that support BPs in a meet-in-the-middle delivery approach.

That is, the main steps of such a process would be as shown in Table 2:

- Specify the architecture of the software as a SOA
- Identify the business functions or pieces of software candidate to be Web services
- Modernize legacy applications that fulfill the requirements
- Construct the services
- Deploy the Web services
- Build SBA as compositions

Table 2. Mapping facet into a process guidance architecture

Facet	Levels of guidance architecture		Delivery Approaches
	Business Level	Technology Level	
Contract	Services from business oriented building blocks	Web services: WSDL	Green-Field or Top-Down
Modernization	Selection of the business functions to modernize for SOA	Wrappers: legacy applications into Web services	Meet-in-the-middle
Deployment		• Web services: construction • Deployment • Management	/
SBAs	BPs architecture	• Orchestrations: BPEL • Choreographies: WS-CDL	Meet-in-the-Middle

4.3 The SBA Process Defined

The process mainly uses a meet-in-the middle delivery approach. It has three phases:

Phase 1: Service contract development

Phase 2: SBA development

Phase 3: Service and SBA monitoring and management.

Each phase is subdivided into steps. Table 3 summarizes the steps of each phase.

Each phase reflects:

- One of the objectives of service-oriented software engineering: service, SBAs, and management and monitoring of both of them.
- Each facet of the MSF is considered in the process.
- Each phase produces an artifact that complies with SOA.
- Each step in the process produces an artifact that complies with SO.
- Each artifact produced in one step is used as input in the next steps.

That is, the main steps of such a process would be as shown in Table 3:

- Decide SOA architecture
- Specify the architecture of the software
- Identify the pieces of software candidate to be Web services

- Wrap legacy applications that fulfill the requirements
- Construct the services
- Deploy the Web services
- Build the composition

Table 3. Mapping the guidance architecture into development process

Facet	Service			SBAs	Management and Monitoring
	Process				
	Identification: Phase 1	Design: Phase 1	Construction: Phase 1	Composition: Phase 2	Deployment and management: Phase 3
Contract	√	√			
Modernization			√		
Deployment				√	
SBAs					√

5. Related Work

More than 50 methods for service and solution engineering have been proposed (Gu & Lago, 2011; Kohlborn et al., 2009). We may classify the approaches to develop these methods into a pragmatic or a more theoretical way. The practical approach concerns with: (i) consolidation of best practices of the existing methods, (ii) extension of an existing method, (iii) analogy with an existing method (e.g., UML), or (iv) product line. The theoretical approach would provide comprehensive framework as an abstract model representing the relevant entities surrounding the paradigm service, their relationships, and their constraints.

In the pragmatic approach, the traditional top-down, and bottom-up methods have been extended to green-field and meet-in-the-middle respectively (Chen & He, 2011) to come up with methods such as SOMA (Arsanjani et al., 2008, SOAF (Erradi et al., 2006), Sensoria (Wirsing et al., 2008), and WS-SOAR (Baghdadi, 2012b). These methods are limited to certain perspectives of services, namely their identification from existing data or functions (Gu &Lago, 2011; Al-Rawahi & Baghdadi, 2005; Kohlborn et al., 2009).

At time of writing, except the work by Erl (2008) who has provided a synthesis of Web services underlying concepts, there is a no meta-model such as the one provided in the object orientation: MOF (Meta Object Framework) of a framework that guides a method for developing services and SBAs.

Indeed, a framework approach would produce several methods, as only one method could not solve all the instances of the problem, even if it is generic, as each enterprise has its own specificity.

The multifaceted MSF is a step toward such a comprehensive theoretical framework.

6. Conclusion

The main contribution is to frame the entities related to service-oriented software engineering that are service science, service orientation, service oriented architecture and service oriented computing concepts into a meta-service framework. This framework is then used to guide service-oriented software engineering methods and processes to develop both services and service-based applications built as service composition solutions.

The resulting process is made up of three phases: (1) service contract development, (2) service-based application composition, and (3) both service and service-based application monitoring and management. Each phase is subdivided into steps.

A framework that considers all the concepts surrounding service-oriented software engineering, the guided method or process will produce services and service-based application that conform to service orientation and SOA.

This work is limited to the IT view of SOA benefits. Indeed, it concerns only with the services and service-based applications as IT components. Business services and business processes as services that represent the business view of SOA will be considered in further work to complete the loop.

This work can be extended by adding the aggregates of a method and the requirements of SOA maturity level to enrich the framework with a comprehensive set of entities, taking into account business and IT views of SOA. Then the produced methods will take into considerations many parameters, including the peculiarities of the methods, the building blocks, and the specifics of each class of problems we want to solve by applying a method.

References

Al-Rawahi, N., & Baghdadi, Y. (2005, June). Approaches to identify and develop Web services as instance of SOA architecture. In Services Systems and Services Management, 2005. Proceedings of ICSSSM'05. 2005 International Conference on (Vol. 1, pp. 579-584). IEEE.

Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Gariapathy, S., & Holley, K. (2008). SOMA: a method for developing service-oriented solutions. *IBM Systems Journal, 47*(3), 377-396. http://dx.doi.org/10.1147/sj.473.0377

Baghdadi, Y. (2007). Specification of a Tool for Monitoring and Managing Web Services Architecture. *In ICEIS, 4*, 51-56).

Baghdadi, Y. (2009, March). A metadata for Web services architecture: A framework for service-oriented software development. In *GCC Conference & Exhibition, 2009 5th IEEE* (pp. 1-6). IEEE.

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), 1-14. http://dx.doi.org/10.1504/IJCAT.2012.050128

Baghdadi, Y. (2012b). A methodology for web services–based SOA realization. *Int. Journal of Business Information Systems, 10*(3), 264-297. http://dx.doi.org/10.1504/IJBIS.2012.047531

Baghdadi, Y., & Al-Bulushi, W. (2013). A guidance process to modernize legacy applications for SOA. *Service Oriented Computing and Applications*, 1-18.

Bianco, P., Kotermanski, R., & Merson, P. (2007). *Evaluating a Service-Oriented Architecture*. CMU/SEI-2007-TR-015.

Brittenham, P. (2001). *Web services development concepts (WSDC 1.0)*. IBM Software Group, USA. http://www.csd.uoc.gr/~hy565/newpage/docs/pdfs/papers/wsdc.pdf

Canfora, G., Fasolina, A. R., Frattolillo, G., & Tramontana, P. (2008). A wrapping approach for migrating legacy system interactive functionalities to service oriented architectures. *The Journal of Systems and Software, 81*(1), 463-480. http://dx.doi.org/10.1016/j.jss.2007.06.006

Chen, H., & He, K. (2011). A Method for service-Oriented Personalized Requirements Analysis. *Journal of Software Engineering and Applications, 4*(1), 59-68. http://dx.doi.org/10.4236/jsea.2011.41007

Comella-Dorda, S., Wallnau, K., Seacord, R. C., & Robert, J. (2000). *A Survey of Legacy System Modernization Approaches*. CMU/SEI-2000-TN-003. Retrieved from http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA377453

Cummins, F. A. (2010). *Building the agile enterprise with SOA, BPM and MBM*. The Morgan Kaufmann/OMG Press.

Erl, T. (2008). *SOA Principles of Service Design*. Prentice Hall, Englewood Cliffs.

Erl, T. (2009). *SOA Design Patterns*. Prentice Hall.

Erradi, A., Anand, S., & Kulkarni, N. (2006, September). SOAF: An architectural framework for service definition and realization. In *Services Computing, 2006. SCC'06. IEEE International Conference on* (pp. 151-158). IEEE.

Gu, Q., & Lago, P. (2011). Guiding the selection of service-oriented software engineering methodologies. *Service Oriented Computing and Applications, 5*(4), 203-223. http://dx.doi.org/10.1007/s11761-011-0080-0

Kohlborn, T., Korthaus, A., Chan, T., & Rosemann, M. (2009). Identification and analysis of business and software services—a consolidated approach. *IEEE Transactions on Services Computing, 2*(1), 50-64. http://dx.doi.org/10.1109/TSC.2009.6

Lewis, G. A., Morris, E. J., Smith, D. B., & Simanta, S. (2008). *Smart: Analyzing the reuse potential of legacy components in a service-oriented architecture environment*. CMU/SEI-2008-TN-008. Retrieved from: http://www.dtic.mil/dtic/tr/fulltext/u2/a489853.pdf

Mazo, R., Salinesi, C., Diaz, D., Djebbi, O., & Lora-Michiels, A. (2012). Constraints: The heart of domain and application engineering in the product lines engineering strategy. *Int. Journal of Information System Modeling and Design, 3*(2), 33-68. http://dx.doi.org/10.4018/jismd.2012040102

Papazoglou, M. P., Andrikopoulos, V., & Benbernou, S. (2011). Managing Evolving Services. *IEEE Software, 2*, 49-55. http://dx.doi.org/10.1109/MS.2011.26

Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2008). Service-Oriented Computing: A Research Roadmap. *Int. Journal of Cooperative Information Systems, 17*(2), 223-255. http://dx.doi.org/10.1142/S0218843008001816

Papazoglou, M. P., & van den Heuvel, W. J. (2006). Service-oriented design and development Methodology. *Int. Journal of Web Engineering and Technology, 2*(4), 412-442. http://dx.doi.org/10.1504/IJWET.2006.010423

Rahgozar, M., & Oroumchian, F. (2003). An effective strategy for legacy systems evolution. *J. of Software Maintenance and Evolution, 15*(5), 325-344. http://dx.doi.org/10.1002/smr.278

Spohrer, J., Anderson, L. C., Pass, N. J., Ager, T., & Gruhl, D. (2008). Service Science. *Journal of Grid Computing, 6*(3), 313-324. http://dx.doi.org/10.1007/s10723-007-9096-2

W3C. (2010). *Web services architecture usage scenarios*. W3C Working Group Note. Retrieved February 11, 2004, from http://www.w3.org/TR/wsarch-scenarios

Welke, R., Hirschheim, R., & Schwarz, A. (2011). Service-Oriented Architecture Maturity. *IEEE Computer, 56*(1), 61-67. http://dx.doi.org/10.1109/MC.2011.56

Wirsing, M., Hölzl, M., Koch, N., Mayer, P., & Schroeder, A. (2008). Service Engineering: The Sensoria Model Driven Approach. *Proceedings of Software Engineering Research, Management and Applications (SERA 2008)*, Prague, Czech Republic.

Copyrights

Copyright for this article is retained by the author(s), with first publication rights granted to the journal.

This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/).