Emerald Insight

International Journal of Web Information Systems

A method engineering perspective for service-oriented system engineering

Boutheina Gherib Youcef Baghdadi Naoufel Kraiem

Article information:

To cite this document:
Boutheina Gherib Youcef Baghdadi Naoufel Kraiem , (2015),"A method engineering perspective for service-oriented system engineering", International Journal of Web Information Systems, Vol. 11 Iss 4 pp. 418 - 441
Permanent link to this document:
http://dx.doi.org/10.1108/IJWIS-03-2015-0004

Downloaded on: 09 November 2016, At: 01:44 (PT)
References: this document contains references to 46 other documents.
To copy this document: permissions@emeraldinsight.com
The fulltext of this document has been downloaded 239 times since 2015*

Users who downloaded this article also downloaded:

(2011),"A query language for selecting, harmonizing, and aggregating heterogeneous XML data", International Journal of Web Information Systems, Vol. 7 Iss 1 pp. 62-99 http://dx.doi.org/10.1108/17440081111125662

(2005),"Adaptation of cultural content: evidence from B2C e-commerce firms", European Journal of Marketing, Vol. 39 Iss 1/2 pp. 71-86 http://dx.doi.org/10.1108/03090560510572025

Access to this document was granted through an Emerald subscription provided by emerald-srm:563821 []

For Authors

If you would like to write for this, or any other Emerald publication, then please use our Emerald for Authors service information about how to choose which publication to write for and submission guidelines are available for all. Please visit www.emeraldinsight.com/authors for more information.

About Emerald www.emeraldinsight.com

Emerald is a global publisher linking research and practice to the benefit of society. The company manages a portfolio of more than 290 journals and over 2,350 books and book series volumes, as well as providing an extensive range of online products and additional customer resources and services.

Emerald is both COUNTER 4 and TRANSFER compliant. The organization is a partner of the Committee on Publication Ethics (COPE) and also works with Portico and the LOCKSS initiative for digital archive preservation.

*Related content and download information correct at time of download.

The current issue and full text archive of this journal is available on Emerald Insight at:
www.emeraldinsight.com/1744-0084.htm

Received 2 March 2015
Revised 12 April 2015
16 April 2015
Accepted 16 April 2015

A method engineering perspective for service-oriented system engineering

Boutheina Gherib
RIADI, Ecole Nationale des Sciences de l'Informatique, Nabeul, Tunisia
Youcef Baghdadi
Department of Computer Science, Sultan Qaboos University, Muscat, Oman, and
Naoufel Kraiem
Department of Computer Science, Sultan Qaboos University, Muscat, Oman and RIADI, Ecole Nationale des Sciences de l'Informatique, Tunis, Tunisia

Abstract

Purpose – The purpose of this paper is to consider the method engineering perspective for service-oriented system engineering (SOSE). A number of SOSE methods have been proposed in both academia and industry. Given this, many intuitive, common questions arise. To answer these questions, many comparison frameworks have been developed. Each of which has considered certain methodological perspectives. However, less attention has been given to the method engineering (ME) perspective. The authors argue that this perspective would answer the question "what ME to apply in order to produce SOSE methods that themselves produce quality services at different levels of abstraction and SBAs". This research question is further decomposed into other questions; the main one is "whether the existing ME do apply to service orientation". Answering such a question would lead to either developing SOSE methods by using the existing ME or developing a new ME or framework for the specifics of SOSE.

Design/methodology/approach – This work first provides a literature review on ME approaches and techniques; then, it compares a sample of existing SOSE methods with respect to their ME within a comparison framework that comprises a set of relevant properties of a solution that would be provided by an SOSE method, namely, service-oriented architecture (SOA) adoption, quality of services and ME; and finally, it discusses the applicability of the existing ME to SOSE.

Findings – Strengths and weaknesses of the existing methods with respect to the aforementioned criteria, in addition to SOSE methodology open issues, were identified. The comparison has shown that while the existing SOSE methods have proved their success in a specific task, they still present some weaknesses. Therefore, it is better to benefit from the advantages of the existing ME techniques, notable method fragments, even if they need some alteration.

Research limitations/implications – While this work has many open issues related to SOSE methods with respect to ME, it could be further developed in many directions by exploring the open issues. For instance, the generation of a new ME technique and application of this new ME technique to the existing SOSE methods to see to what extent the existing methods may be situational.

International Journal of Web Information Systems
Vol. 11 No. 4, 2015
pp. 418-441
© Emerald Group Publishing Limited
1744-0084
DOI 10.1108/IJWIS-03-2015-0004

This work is a part of a project funded by The Research Council (TRC) of the Sultanate of Oman, under Grant No: ORG/SQU/ICT/14/004 (www.trc.gov.om).

Practical implications – This work has practical implications, as it provides a better understanding of different views of SOSE methods, and assists the method engineers in deciding which ME technique is most suitable to their situation.

Social implications – The produced artifact provides a research roadmap toward SOSE ME.

Originality/value – None of the existing comparison frameworks for SOSE methods has considered the criteria such as SOA adoption and ME techniques. Indeed, ME techniques and approaches would allow better reuse of the existing proven fragments of methods.

Keywords Advanced Web applications, E-business models and architectures, Internet quality of service, Web design metrics

Paper type Research paper

1. Introduction

Service-oriented architecture (SOA) is an architectural style that promotes flexibility and agility through abstraction, separation of concerns, loose coupling and interoperability of its basic components that are services. What makes SOA attractive is the ability to create service-based applications (SBAs) by composing existing services. Web services (WSs) technology constitutes a suitable distributed computing platform to realize SOA (Baghdadi, 2012a), as WSs have features such as interoperability and loose coupling, which make them better in composition inside the heterogeneous environments.

SBA architecture has three main components: providers, consumers and a registry. Providers publish or announce their services on registries where consumers can find and then invoke them (Baghdadi, 2013). The Enterprise Service Bus (ESB) is responsible for routing, controlling and transforming the communication between providers and consumers. It can be understood as a layer, which is adding value by providing various functions, so far provided by the requesters and providers (Meier, 2006).

However, developing SBAs with respect to SOA requires an approach to address systematic, disciplined, quantifiable methods (Gu and Lago, 2009). The quantitative paradigm is based on positivism. Techniques to ensure this include randomization, blinding, highly structured protocols and written or orally administered questionnaires with a limited range of predetermined responses (Sale *et al.*, 2002).

Such an approach would address engineering of services, engineering of compositions and quality of the products that are services and compositions (Papazoglou *et al.*, 2011) in terms of a set of common activities performed in a process model such as top-down, bottom-up, meet-in-the middle or green-field (Baghdadi, 2012b). These activities are service identification, service realization (contract specification and wrapping or implementation) and service composition. The service identification concerns with identifying candidate services that are logically coherent, independent and reusable. The service realization concerns with defining service contracts as well as service implementation, testing and deployment; subsequently, the service composition concerns with composing services into SBAs that support business processes (Baghdadi, 2006a). We refer to this approach as service-oriented software engineering (SOSE).

A number of SOSE methods have been proposed in both academia and industry (Al-Rawahi and Baghdadi, 2005; Gu and Lago, 2011), such as service-oriented modeling and architecture (SOMA) (Arsanjani *et al.*, 2008), service-oriented analysis and design (SOAD) (Zimmermann *et al.*, 2004), service-oriented unified process (SOUP) (Mittal, 2006), the service orientation (SO) process by component-based development and integration (CBDI) (Allen, 2007) or the reverse engineering (Baghdadi, 2006b). Given these number of methods, many intuitive, common questions arise, such as what is the best method that users can choose to suit most of their needs. To answer these questions, many comparison frameworks have been developed (Baghdadi, 2013; Gholami *et al.*, 2010; Gu and Lago, 2010; Gu and Lago, 2011; Kontogogos and Avgeriou, 2009; Ramollari *et al.*, 2007; Rosane *et al.*, 2014). Each of which has considered certain methodological perspectives.

However, less attention has been given to the method engineering (ME) perspective. We argue that this perspective would answer the question "which ME to apply in order to produce SOSE methods that themselves produce quality services (at different levels of abstraction) and SBAs?" This research question is further decomposed into other questions; the main one is "whether the existing ME do apply to service orientation". Answering such a question would lead to either developing SOSE methods by using the existing ME or developing a new ME or framework for the specifics of SOSE.

This work first provides a literature review on ME; then, it compares a sample of existing SOSE methods, in both academia and industry, with respect to their engineering method. This work will assist both the users and developers to choose the most appropriate SOSE method that best suits their needs. This choice will help developers either to create a new SOSE method or to use an SOSE method just in the aim to generate an SBA. Finally, we discuss the applicability of the existing ME to SOSE.

The remainder of this paper is organized as follows: Section 2 presents the definition of some concepts used in this paper and introduces the concept of ME. Section 3 summarizes the most popular existing SOSE methods in the literature. In Section 4, we detail the comparison framework within which we compare the selected SOSE methods. In Section 5, we discuss the comparison. In Section 6, we expose some related work. Finally, Section 7 concludes the paper and presents some directions for future work.

2. Method engineering

In this section, we summarize the engineering approaches and the aggregates any method should have.

2.1 Terminology

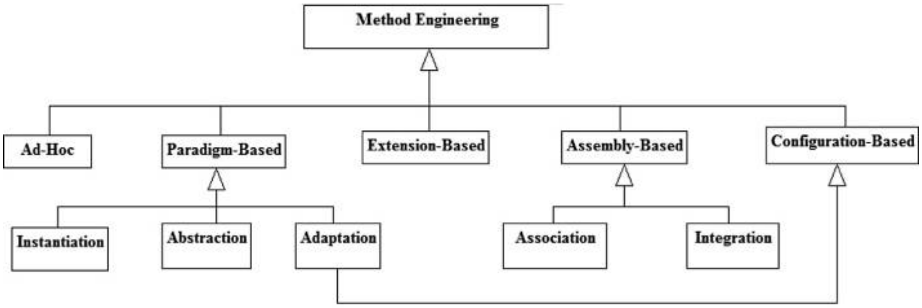
In this section, we define some concepts which are used in this paper, namely, the terms framework, method, approach, technique and ME:

- A framework is a set of entities and relationship between entities and a set of constraints. It is an abstract model that helps understanding a reality. It can be further developed to generate methods.
- A method uses a process to develop a product (software). It is made up of a set of aggregates, namely, representation techniques (e.g. models, formalisms, languages) and tools that assist the developer.
- An approach is a specific way of thinking that consists of directions and rules, structured in a systematic way in development activities with corresponding development products (Brinkkemper, 1996).
- A technique is a procedure, possibly with a prescribed notation, to perform a development activity (Brinkkemper, 1996).
- An ME is an approach or a technique used to develop a new method for a specific context or project.

2.2 Engineering approaches

A large number of ME approaches have been proposed in the literature. These approaches are classified by Ralyté *et al.* (2004) into four types: *ad hoc*, paradigm-based, extension-based and assembly-based. In addition, and based on the work of Karlsson and Gerfalk (2004), there exists also the method configuration which can be considered (Figure 1):

- *Ad hoc approach*: This approach deals with the construction of a new method from scratch. There are different reasons to construct a new method. The appearance of a new application domain that is not yet supported by a method is one example of such reasons. The *ad hoc* approach is characterized by a situation in which project management figures are difficult to measure, few guidelines for project management and information system (IS) modeling procedures are available and project resources such as people and money are allocated on an *ad hoc* basis. Moreover, there is no uniform terminology available among IS projects, and sometimes not even among individual project team members. There is little guidance, and project managers and team members are relying on their experience and capabilities, which makes project success, to a large extent, dependent on the people that make up a project. Systems engineering data are not centrally stored, but distributed over people's workstations or, which is even worse, heads (Harmsen, 1997).
- *Paradigm-based approach*: This approach uses some initial paradigm model or meta-model as a baseline *as-is* model to develop the new *to-be* model. In this approach, an existing meta-model is adopted, instantiated or abstracted to create a new method, according to the current ME objective.
- *Extension-based approach*: This approach aims at enhancing a method with new concepts and properties. For example, a static method to construct an E/R schema can be extended to deal more systematically with the representation of time through a calendar of time points.
- *Assembly-based approach*: This approach proposes to construct new methods or to enhance existing ones by reusing the parts of other methods. The main idea is the reuse of method components (aka method fragments or method blocks). An assembly-based method construction consists in first defining method requirements of a current situation, then selecting method components satisfying this situation and finally assembling them.



Source: Adapted from Ralyté *et al.* (2004)

Figure 1.
Typology of ME approaches

- *Configuration-based approach*: This approach aims to adapt a particular method to various situated factors. This type of ME configures one existing method rather than assembling a set of methods. In addition, this method configuration can also be enhanced with additional fragments from other methods. Consequently, we consider method adaption as a kind of configuration.

In addition to these approaches, we also found the following techniques:

- *Using the computer-aided method engineering (CAME) tool*: CAME practice is a disciplined process to build, evaluate or modify a method by means of component specification method and the relationship between them (Ralyté, 2001).
- *Using generic patterns*: A pattern provides a reusable solution in any situation where the problem concerned with the pattern appears. With its ability to reuse, this concept has recently been introduced in the field of engineering methods (Ralyté, 2001).
- *Product line*: This approach aims at consolidating key software assets within a high-quality, reusable software core, and concentrate their resources on adapting this core to meet the changing needs of customers (Catkinson *et al.*, 2000).

2.3 Method aggregates

Engineering methods for SOSE depends on many parameters, including the main building blocks, the state of the art, the organization practices and skills and the nature of the problems to be solved, to cite a few. These parameters vary from method-to-method, which explains the existence in the market of a number of methods around the same computing paradigm and the same nature of problems. Yet, most of the methods have a sound set of invariant aggregates that we can represent in a meta-model, where each distinct instantiation of the meta-model produces a method. These aggregates are:

- process activities;
- representation techniques, including models, languages, formalisms, diagrams and notations;
- tools; and
- inspection techniques, as shown in Figure 2.

The process is a set of coordinated engineering activities that produce a product made up of:

- views of the solution at different levels of abstraction/refinement with respect to different types of stakeholders; and
- documentation of the product, including documentation of all the views.

The representation techniques are models, formalisms or languages, along with notations and diagrams, used to represent and express the views to reduce their complexity and make them understandable by their respective stakeholders.

The tools, namely, computer-aided software engineering (CASE) tools, assist in producing the views.

The inspection techniques assess and evaluate the quality of each view of the solution. These are the metrics that concern with the quality of the services, views and solutions.

In addition to the aggregates, an SOSE method:

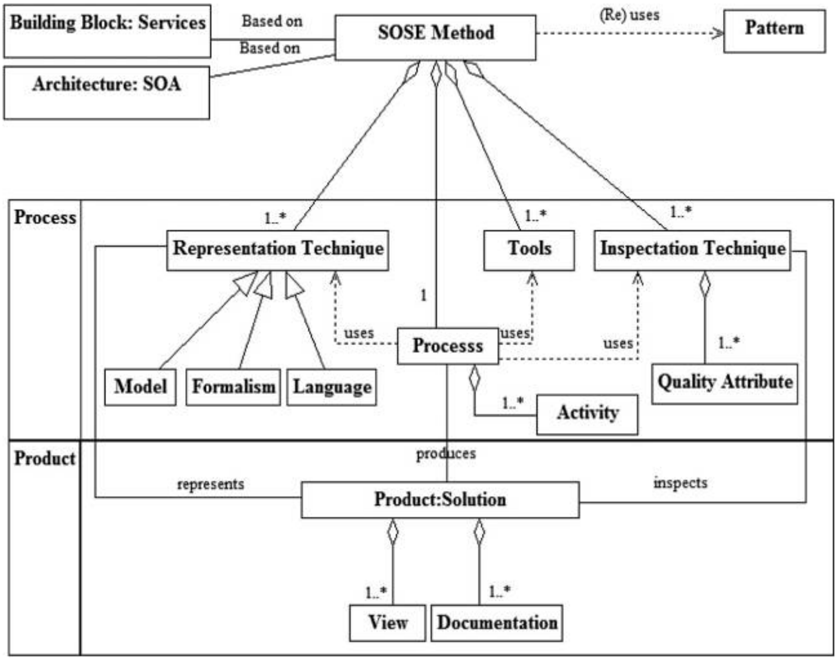

Building Block: Services
Based on
Based on
Architecture: SOA
SOSE Method
(Re) uses
Pattern
Process
1..*
Representation Technique
1..*
Tools
1..*
Inspectation Technique
1
uses
Processs
uses
uses
1..*
Model
Formalism
Language
1..*
Activity
Quality Attribute
Product
produces
represents
Product:Solution
inspects
1..*
1..*
View
Documentation

Figure 2.
A model specifying the invariant, static aspects of SOSE methods (Baghdadi, 2015)

- Is based on service and SOA, as service is the main computational building block having a set of fundamental properties and principles provided by service science and SO, respectively. SOA shapes the architecture of the produced solution as a composition of services. That is, the architecture of the solution should comply with the principles of SOA, and is distributed with respect to service-oriented computing paradigm, a realization infrastructure having its underlying technologies and standards provided by the Web.
- Re-uses patterns, well-known aggregates such as process, representation techniques, fragments of methods or inspection techniques.

3. Overview of SOSE methods

In this section, we first summarize the most known methods among the 24 methods we have studied (Alani and Baghdadi, 2012) and the 60 methods that exist in the market (Gu and Lago, 2011). The following methods have been selected with respect to the availability of documentation and their previous use in different existing comparison frameworks.

3.1 Method 1 (M1): SOMA

SOMA is a software development life cycle method invented and initially developed by IBM for designing and building SOA-based solutions. It consists of three steps:

(1) identification;

(2) specification; and

(3) realization of services flows (business processes) and components realizing services.

The process is highly iterative and incremental (Arsanjani *et al.*, 2008). Concerning the aggregates of methods, SOMA provides developers with the rational unified process (RUP) that can be followed to obtain SBA (Arsanjani *et al.*, 2008).

3.2 Method 2 (M2): Service-oriented architecture framework (SOAF)

SOAF provides a systematic approach and a well-defined process to guide the design, evaluation and development of SOA. This method contains five main phases:

1. information elicitation;
2. service identification;
3. service definition;
4. service realization; and
5. roadmap and planning (Erradi *et al.*, 2006).

Concerning the aggregates of methods used by SOAF, it uses the business process modeling (BPM) to extract the candidate services (Erradi *et al.*, 2006).

3.3 Method 3 (M3): SOAD

The interdisciplinary SOAD method combines techniques such as object-oriented analysis and design (OOAD), enterprise architecture (EA) frameworks and BPM to facilitate successful SOA deployments. SOAD proposes a meet in the middle approach for identifying services. In addition to the use of OOAD, BPM and EA, the SOAD method also uses the unified modeling language (UML) to extract the candidate services (Zimmermann *et al.*, 2004).

3.4 Method 4 (M4): SOUP

As the name suggests, this approach is primarily based on the RUP. Its life cycle consists of six phases: incept, define, design, construct, deploy and support. SOUP methodology can be used in two slightly different variations: the first one adopts RUP for initial SOA projects, whereas the second adopts a mix of RUP and extreme programming (XP) for the maintenance of the existing SOA applications (Mittal, 2006).

3.5 Method 5 (M5): The SOAD method by Erl

The SOAD methodology documented in Thomas Erl's book (Erl, 2005) is considered to be the first vendor-agnostic one to be published. This methodology is a step-by- step guide through the two main phases: analysis and design. Service-oriented analysis comprises three main steps:

1. define business requirements;
2. identify existing automation systems; and
3. model candidate services.

Service-oriented analysis results in the preparation of "to-be" process model that SOA application will implement. The activities in the analysis phase take a top-down business view where service candidates are identified. These serve as input for the next phase, service-oriented design, where the service candidates are specified in detail and later realized as WSs (Erl, 2005).

3.6 Method 6 (M6): Web services development life cycle methodology (SDLC)

The SDLC methodology covers the whole SOA life cycle. It is partly based on well-established development methodologies as RUP, component-based development (CBD) and BPM. The methodology is based on iterative and incremental process and comprises one preparatory planning and eight main phases: service analysis, design, construction, test, provisioning, deployment, execution and monitoring (Papazoglou and van den Heuvel, 2006).

3.7 Method 7 (M7): SENSORIA development approach

SENSORIA presents an approach for service engineering called SENSORIA development approach (SDA). The SDA is intended to support model-driven engineering by including formal methods and tools at the appropriate steps, thus building a formally underpinned development approach. SDA is supported by tools that are integrated into the common integration platform (Wirsing *et al.*, 2008a, 2008b).

3.8 Method 8 (M8): A consolidated approach for identification and analysis of business and services

The integrated, consolidated approach (aka identification and analysis of business and software services) provides an SOSE method for service identification, which comprises seven stages: the first three stages, namely, preparation phase, identification phase and detailing phase, are concerned with defining business services. A prioritization phase in which services that are suitable for realization are chosen and existing application systems are analyzed. The last three phases are concerned with defining software services. Each stage is further decomposed into a set of activities (Kohlborn *et al.*, 2009).

3.9 Method 9 (M9): CBDI-SAE SOA reference framework

The service architecture and engineering (SAE) reference framework is designed to provide a comprehensive framework of all the components that are necessary to support the migration to a service-oriented enterprise. This methodology aims at integrating business–information technology (IT) through top-down analysis of business requirements as well as the bottom-up legacy system. The CBDI-SAE process aims to cover the whole SOA life cycle, including deployment, monitoring and governance activities (Allen, 2007).

4. The proposed comparison framework

Traditionally, we use an ME approach to produce a new method. This approach has proven useful for engineering methods based on paradigms such as function, object or component as the main building block (or construct). However, the issue now is the pertinence of the existing ME approaches to produce methods that themselves produce quality services and SBAs. To our knowledge, the existing ME approaches concern only with limited perspectives and aspects of SO (Gholami *et al.*, 2011). Therefore, we propose a framework within which we compare the above-mentioned service-oriented methods, where we adopt a faceted classification approach (Selmi *et al.*, 2005).

The framework consists of four views as shown in Figure 3. Each view is associated with a set of facets. These facets constitute viewpoints or dimensions suitable to characterize and classify SOSE methods according to this view (Selmi *et al.*, 2005). A metric is attached to each facet which is measured by a set of relevant attributes.

SOSE methods are positioned in the framework by affecting values to the attributes for each facet. Attribute values are defined within a domain which may be a predefined type (Integer, Boolean, etc.), an enumerated type (ENUM{x, y, z}) or a structured type (SET or TUPLE) (Selmi *et al.*, 2005).

4.1 The views

The views in the framework consider four comparison aspects:

1. the ME techniques (if any) used to engineer (produce) the method;
2. the aggregates used in the method;
3. the purpose of the method; and
4. the quality of the generated (produced) services.

4.1.1 View 1: ME techniques. The ME technique view shows to what extent the production of the existing SOSE methods has followed the existing ME techniques.

The ME technique view is characterized by one facet, namely, *the ME approaches used*. Indeed, to create a new method, we usually use one of the ME approaches such as *ad hoc*, paradigm-based ME, extension-based ME, assembly-based ME and product line. The values of the attribute for this facet are: ME approach: (Enum: "*Ad hoc*", "Paradigm-Based", "Extension-Based", "Assembly-Based", "Product line").

4.1.2 View 2: Method aggregates. The aggregate view shows to what extent the existing SOSE methods use the different generic aggregates (a method should have), namely, process, product, tools and techniques. Indeed, we expect the aggregate to be a fragment or composition of fragments.

The method aggregate view is characterized by one facet, namely, the *used aggregates*. The use of the aggregates differs from one SOSE method to another. The SOSE methods may use UML (e.g. SOAD); RUP (e.g. SOMA, SOUP and SDLC); or OOAD, BPM and CBD (e.g. SOAD, Erl's method, SDLC). Also the aggregates of a method are not used in all the phases, but only in specific cases and for a specific purpose. The values of the attribute for this facet are: *Aggregate used*: (Enum: Process, Product, Tool, Technique).

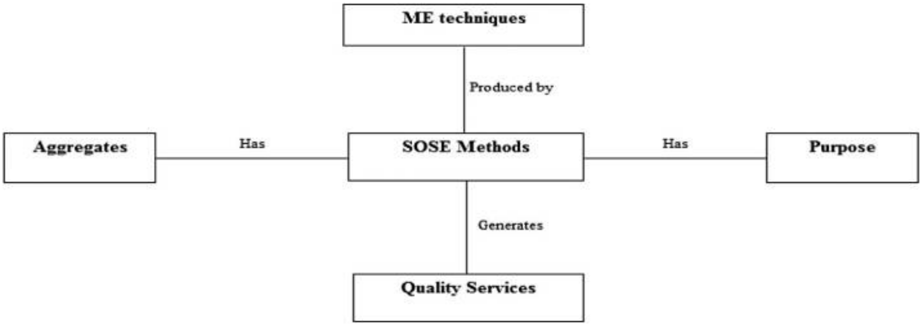


Figure 3.
The four views of the proposed comparison framework

4.1.3 View 3: Purpose view. The purpose view deals with intentional aspects. It concerns the services and SBAs generated by the SOSE method. The service-oriented solutions are based on SOA as an architectural style, but the conversion to an SOA is not immediate. Instead, it depends on the legacy application and it goes through different steps, as we will show below. Therefore, this view concerns with the level of SOA maturity of the application generated and also the types of services generated. It has two facets: *level of SOA maturity* and *type of the generated services* (e.g. technology services, business services or compositions).

4.1.3.1 Facet 1: SOA maturity. When adopting SOA, different areas in a company are affected (Pereira and Sousa, 2004): organizational structure, people, workflow processes and technologies. Adopting SOA is not easy and a lot of challenges arise, e.g. immature standards and insufficient knowledge (Tilley *et al.*, 2004). So, the migration from a legacy application to SOA application is neither easy nor direct. In fact, it goes through several steps. For that, there are five levels of SOA maturity, which are:

1. *Level 1* (*L1*): Initial. It is the basic level of SOA maturity where the thought of SOA has just entered the designer's mindset (Inaganti and Aravamudan, 2007). At the initial stage, most firms appear to view SOA from the more limited IT-perspective as fine-grained software components. They have not yet adopted any form of SOA governance. Also, they tend to use a somewhat loose definition of services with a minimum of change (Welke *et al.*, 2011).
2. *Level 2* (*L2*): Managed. It is a level of SOA maturity where services are engineered to be loosely coupled. IT costs begin to show a return on investment, and multiple applications within the enterprise are now integrated using open standards with a common middleware such as ESB (Inaganti and Aravamudan, 2007). The services are still defined relative to internal IT needs. However, they have shifted from application programming interfaces and wrappers around existing functionality to an active definition and development of new services (Welke *et al.*, 2011).
3. Level *3* (*L3*): Defined. It is a level of maturity that can be reached when services are self-contained and fine-grained enough to function as an independent service and yet flexible enough to be a part of process orchestration. In the previous level, service identification was done using a bottom-up approach from the application portfolio. This level sees an introduction of the top-down method of service identification from the business level, for better business alignment. BPM tools, business orchestration servers and business process rules would be introduced at this level. A mechanism of discovering existing services for reuse, such as service registry, would be used. This would enable business processes to change quickly and effectively (Inaganti and Aravamudan, 2007).
4. *Level 4* (*L4*): Quantitatively managed. It is a level of SOA maturity where composite business services are measured and fine-tuned for better performance, flexibility and re-use (Inaganti and Aravamudan, 2007). The business is now brought fully in the SOA approach and a shift occurs to "service thinking", as IT and business governance metrics are now aligned (Welke *et al.*, 2011).
5. *Level 5* (*L5*): Optimized. It is the final level of SOA maturity. The enterprise's optimized services are dynamically reconfigurable. Also, it would automatically sense and respond to change in service delivery of business

processes during the run-time. Services and processes would automatically change their behavior not only based on IT performance metrics but also on business performance metrics. At this level, the SOA becomes fully optimized and aligned with business (Inaganti and Aravamudan, 2007). This level of maturity will bring firms the benefits both at the enterprise and IT level (Welke *et al.*, 2011).

4.1.3.2 Facet 2: Types of generated services. The service-oriented solutions are constructed from services. These services are of different types. In fact, many researches (Erl, 2005; Kohlborn *et al.*, 2009; Gu and Lago, 2010; Cohen, 2007) have been done to classify these services. These types are compiled from (Alani and Baghdadi, 2012; Lago and Razavian, 2012):

- conceptual services (agnostic to implementation);
- communication services (expose message transfer capabilities such as message-routing);
- auxiliary services (provide facilities for monitoring, diagnostics and management activities of other services);
- task/activity services (expose action-centric business logic elements);
- process services (compose and orchestrate other application layer services to implement business process);
- data/entity services (operations on data/business-centric entities/data);
- utility services (expose the functionality of an application, including migrated;
- (re-)engineered legacy systems, or some commercial of the shelf software;
- hybrid services (contain both application and business logic functionalities, including wrapped legacy systems, and components); and
- external services (offered by an external party).

The values of the attribute for these two facets are:

- *Level of SOA maturity*: (Enum: L1, L2, L3, L4, L5).
- *Types of services*: (Enum: Technology Services [TS], Business Services [BS], Compositions [C]).

4.1.4 View 4: Quality of generated services. The quality view gives an idea on the products of the existing methods. It is characterized by two facets, namely, the quality *of services* facet and the *quality of SBAs* facet.

4.1.4.1 Facet 1: Quality of services. This facet introduces the quality of services that an SOSE method can generate. Indeed, the non-functional properties of services include temporal and spatial availability, channels, charging styles, settlement models, settlement contracts, payment, service quality, security, trust and ownership (O'Sullivan *et al.*, 2002). Therefore, we have chosen three possible attributes, which are:

(1) *None*: If the SOSE method does not respect any of the quality of services mentioned above.

(2) *To some extent*: If the SOSE method uses some of the quality of services mentioned above.

(3) *Most*: If the SOSE method uses most or all of the quality of services mentioned above.

4.1.4.2 Facet 2: Quality of SBAs. This facet concerns with the quality of SBAs. Indeed, in addition to SBA functional requirements, performance and quality of service deserve a special attention. Performance and quality of service for SBAs include response time, throughput, reliability, safety, security and availability. We have chosen three possible attributes:

1. *None*: If the SOSE method does not respect any of the quality of SBAs mentioned above.
2. *To some extent*: If the SOSE method uses some of the quality of SBAs mentioned above.
3. *Most*: If the SOSE method uses most or all of the quality of SBAs mentioned above.

So, this criterion introduces the quality of SBA that an SOSE method can generate.

The values of the attributes for these two facets are:

- *Quality of services*: (Enum: None, To some extent, Most).
- *Quality of SBAs*: (Enum: None, To some extent, Most).

4.2 Comparison

The following tables summarize the comparison of the existing SOSE methods within the above developed framework.

Table I shows which of the ME techniques have been followed by each of the existing SOSE methods, where "✓" means that the method is produced by using one of the existing ME.

Table II shows the generic aggregates used by each SOSE method, where:

1. *Process*: Indicates the process used by the SOSE method and it can take three possible values:
    - *The name of the process used (RUP, XP)*: When the SOSE method indicates clearly the name.
    - *Phases + activities*: When the SOSE method does not indicate the name of the process; however, it proposes a detailed guideline to be continued to obtain SBA.
    - *Steps*: When the SOSE method does not indicate the name of the process; however, it has just proposed some steps and not a detailed guideline.
2. *Product*: Indicates what does the SOSE method produce and it can take two values:
    - *SBA*: When the SOSE method produces SBA.
    - *Services*: When the SOSE method just produces services.
3. *Tool*: Indicates the tools used by the SOSE method and it can take two values:
    - The name of the tool used (PAM, UML, UML4SOA, Pepa, WS-Engineer).

Table I.
ME approaches used by SOSE methods

				ME approaches		
SOSE methods		Ad hoc	Paradigm-based ME	Extension-based ME	Assembly-based ME	Product line
M1	SOMA	✓				
M2	SOAF	✓				
M3	SOAD					✓
M4	SOUP			✓		
M5	Erl	✓				
M6	SDLC					✓
M7	SENSORIA		✓			
M8	Consolidated approach				✓	
M9	CBDI-SAE	✓				

Table II.
Aggregates of method used in SOSE methods

			Method aggregates		
SOSE methods		Process	Product	Tool	Technique
M1	SOMA	RUP	SBAs	NO	NO
M2	SOAF	Phases + activities	SBAs	PAM	BPM
M3	SOAD	Steps	Services	UML	OOAD BPM EA
M4	SOUP	RUP XP	SBAs	UML	NO
M5	Erl	Phases + activities	SBAs	NO	BPM
M6	SDLC	RUP	SBAs	NO	CBD BPM
M7	SENSORIA	Phases + activities	SBAs	UML4SOA Pepa WS-Engineer	NO
M8	Consolidated approach	Steps	Services	NO	NO
M9	CBDI-SAE	Phases + activities	SBAs	UML	NO

- *NO*: When there is no information available if the SOSE method uses a tool.

(4) *Technique*: Indicates the techniques used by the SOSE method and it can take two values:

- The name of the technique used (BPM, OOAD, EA and CBD).
- *NO*: When there is no information available if the SOSE method uses a technique.

Table III shows the level of SOA maturity, where "✓" indicates the level of maturity reached when the method is applied, and the table shows also the types of services generated by the SOSE method, where "✓" indicates the type of services the method produces: TS, BS or C.

Table IV presents the quality of services and the quality of SBAs generated by the SOSE method, where "✓" indicates to what extent the method is interested by both the quality of services generated and the quality of SBAs generated.

5. Discussion

In this section, we discuss successively the existing SOSE methods against each view of the framework.

5.1 SOSE methods and ME techniques

SOMA is created from scratch, as this method was harvested from hundreds of successful experiences and lessons learned from the difficulties and challenges encountered in early SOA design and implementation projects. So, the developers of SOMA have not started from a model or an existing method but from the initial experiences in different domains to apply the SOA design and implementation in their projects.

SOAF uses an *ad hoc* approach, as it offers the best practices and guidelines to follow, with the aim to obtain an SBA. It does not rely on any existing paradigm or method.

SOAD not only combines existing elements such as OOAD, EA frameworks and BPM but also add new elements to obtain a method, so it uses the technique of product line.

SOUP uses the best elements from RUP and XP to build and manage SOA project, so it is clear to conclude that this method uses an extended technique.

SOSE methods		Level of SOA maturity					Types of services		
		L1	L2	L3	L4	L5	TS	BS	C
M1	SOMA								✓
M2	SOAF		✓					✓	
M3	SOAD		✓					✓	
M4	SOUP		✓					✓	
M5	Erl		✓					✓	✓
M6	SDLC			✓				✓	
M7	SENSORIA		✓					✓	
M8	Consolidated approach		✓					✓	
M9	CBDI-SAE			✓				✓	

Table III. Level of SOA maturity and the types of services generated by the SOSE methods

SOSE methods		Quality of services			Quality of SBAs		
		None	To some extent	Most	None	To some extent	Most
M1	SOMA		✓		✓		
M2	SOAF		✓		✓		
M3	SOAD	✓			✓		
M4	SOUP	✓			✓		
M5	Erl	✓			✓		
M6	SDLC		✓		✓		
M7	SENSORIA	✓			✓		
M8	Consolidated approach	✓			✓		
M9	CBDI-SAE	✓			✓		

Table IV. The quality of services and the quality of SBAs generated by the SOSE methods

Erl's method does not use any existing paradigm or method, so the ME approach used is the *ad hoc* one.

SDLC method uses a product line approach, as it is partly based on other successful related development models such as RUP, CBD and BPM.

SENSORIA uses the paradigm-based ME technique, as SDA follows the patterns described in the OMG's model-driven architecture, which is an approach that uses models in software development.

The consolidated approach evaluates 30 existing service-oriented methods to investigate their weaknesses and their strengths and then combines and extends the strengths of the different examined methods, so it is clear that this method has used the assembly technique.

CBDI-SAE process method has used an *ad hoc* approach, as it does not use any existing method or paradigm, but it uses a framework that guides the developer to obtain SBA. CBDI-SAE process provides a framework which allows the developers to obtain the SBA; the framework helps the engineer by providing the best practices and guidelines to obtain the desired result.

5.2 SOSE methods and aggregates

SOMA provides developers with a process that can be followed to obtain SBA. However, it does not provide any tool or technique that makes SOMA more understandable for developers.

SOAF uses the process-to-application mapping (PAM), which provides the basis for identifying applications that support a particular business process which are identified by the BPM. In addition, it proposes a detailed guideline that users can follow to obtain SBA.

SOAD uses UML in the application level to extract the different classes and objects that will be then transformed to services into SBAs.

SOUP is based on two processes that are RUP and XP. In addition, it uses UML, as the services which will be built are identified based on use-cases.

Erl's method uses the BPM in extracting the candidate services from the business process of application wanted to be an SBA.

SDLC method is based on RUP process and it uses the two techniques which are CBD and BPM.

SENSORIA uses several tools and techniques to obtain the desired SBA.

The consolidated approach is the result of an analysis of 30 existing service-oriented methods, so this method proposes different steps which users can follow to obtain services. But, it does not use any technique or tool.

CBDI-SAE process method provides a framework which allows the developers to obtain the SBA; the framework helps the engineer by providing the best practices and guidelines to obtain the desired result. In addition, this method uses UML in the implementation view to capture the units that implement the service identified in the specification phase.

5.3 SOSE method purpose

By using SOMA, we could reach the Level 3 of SOA maturity. SOMA has a fractal software development life cycle. It contains several phases and each phase holds activities. SOA maturity and transformation roadmap is considered as an activity at the third phase, business modeling and transformation. In addition, the identification of services is based on the existing business process to realize new services, which corresponds to the Level 3 of SOA maturity, where the concept of reuse of the existing business process is introduced. Concerning the service engineering, the SOMA method generates compositions services.

SOAF has no clear activities that deal with SOA maturity. However, we could reach Level 2 of SOA maturity. Concerning the service engineering, the SOAF method identifies services based on the business process and existing application.

SOAD proposes to reach Level 2 of SOA maturity, as it is interested principally in the analysis and design of services to migrate to SOA. Concerning the service engineering, this method generates services based on business process and legacy application.

SOUP method could reach Level 2 of SOA. The services are defined, but these services are not fine-grained enough to function as an independent service. Concerning the service engineering, the SOUP method creates services based on use-cases and existing application.

Erl's method proposes different phases, but it neither interested nor showing which targeted SOA maturity level. Concerning the service engineering, Erl's method creates services based on legacy application; and also composes new services based on new requirements.

By applying the SDLC method, one can reach the Level 3 of SOA maturity. Indeed, this method covers the whole life cycle of SOA development, so the BPM tools, business orchestration servers and business process rules are all covered by this method. Concerning the service engineering, this method creates services based on legacy application.

SENSORIA method does not consider the maturity of the SOA produced. However, we could expect it to reach the Level 2 of SOA maturity. Concerning the service engineering, this method is also based on the business process and legacy application to create services. SENSORIA uses several tools and techniques to obtain the SBA.

The consolidated approach could reach the Level 2 of SOA maturity. This method is created by studying 30 of the existing SOSE methods, so the authors are not interested in the SOA maturity. Concerning the service engineering, to create services, this method uses legacy application and business process.

The CBDI-SAE method targets the Level 3 of SOA maturity. In fact, this method is a framework that guides the user to create SBAs. In this framework, the BPM tools and business orchestration servers are introduced. Concerning the service engineering, this method creates service based on business process.

5.4 SOSE methods and quality of services

SOMA method proposes different phases that developers can follow to obtain SBAs. These phases generate services. Yet, this method is not interested by the quality of SBAs.

SOAF method proposes a framework that guides developers to produce SBAs. In this framework, some of the qualities of services are covered such as security, but the quality of SBAs is not studied.

SOAD is neither interested in the quality of services, nor by the quality of the SBAs generated.

SOUP method proposes six phases. In these phases, the quality of services and the quality of SBA are not covered.

Erl's method has not been concerned with the quality of services, nor by the quality of the SBAs.

SDLC method proposes several steps. In the service policy concerns of the design phase, the quality of services generated is covered. Quality of services issues that include service reliability, scalability and availability are studied. But in this method, the quality of SBAs is not studied.

SENSORIA method is not concerned with the quality of services generated and also it does not cover the quality of the SBAs generated.

The consolidated approach proposes a new SOSE method after an analysis of 30 existing SOSE methods. So, this method tries to avoid the limitations of other SOSE methods. Yet, it has not been concerned with the quality of services, nor the quality of SBAs generated.

CBDI-SAE method neither covers the quality of services, nor the quality of the SBAs.

5.5 Summary of the comparison

The service-oriented approach promotes the reuse of existing assets in the development of new services and represents an effective interoperability solution in distributed and heterogeneous environments. Despite these advantages, the engineering of services and SBAs faces several challenges, specifically requirement specification, identification, discovery, deployment, composition, integration of services and, last but not least, testing and inspection. To address these topics, several software engineering methodologies and related tools have been proposed in the service-oriented domain. Some of these cover the whole application life cycle (from requirements to testing), whereas others address specific aspects (Gholami *et al.*, 2011).

The SBAs are specific applications that have different characteristics compared to traditional applications. In addition, SBAs are composed of services from different typology. So, existing methodologies cannot be used in their current state. For that, to generate SBAs, we usually use SOSE methods. However, existing SOSE methodologies have the same underlying assumptions as traditional software engineering methodologies. These methods assume that the developer is the owner of the services, and has a system model in mind, carrying holistic definition of all functionalities that should be produced. Both assumptions are of course not true, as in SO, services are neither owned nor part of a monolithic system. The SO paradigm introduced a shift in the way in which an application is conceived. Therefore, SOSE methods should be in-line with such a shift and cover some essential ingredients (Lago and Razavian, 2012):

- *Support for engineering both services and SBAs*: Developers of service-oriented methods can play the roles of service provider, service consumer or both.
- *Focus on SOA*: SOA is an architectural style that supports SO. As such, it should support mechanisms for service publication, dynamic discovery and composition.
- *Embrace the "Open World Assumption"*: Developing SBAs or service compositions means that reuse is planned at design time, but can be actually tested only at run-time. Modern SOSE methods must hence support a mix of design and run-time development activities to make sure that dynamic aspect is well-engineered and the resulting software is reliable.

Based on our comparison framework, we validate the point of view of both Gholami *et al.* (2011) and Lago and Razavian (2012), whereby we can extract the following remarks:

- The SOSE methods studied are created either from scratch or they have used an existing ME approach. So, we can conclude that none of the existing SOSE methods has considered the specifics of the generated application. However, they have just used existing ME approaches and try to apply them in the domain of SO.

- All the SOSE methods try to propose a process that users can follow to obtain either services only or SBAs. The process proposed is different from a SOSE method to another and it differs in the amount of detail which it provides.
- The SOSE methods are based on the UML, BPM and CBD. These different techniques are used either to extract services or to model identified services.
- Most of the existing SOSE methods are not yet interested by the level of SOA maturity in their development.
- The types of services generated by the SOSE methods are business services.
- All the SOSE methods studied are interested in the quality of services generated, but, none of the existing SOSE methods has studied the quality of SBAs generated.

To fit with these changes and in-line with the SO development, significant efforts are required. These efforts are focused on customization of existing methodologies or on definition of new ones without any fruitful reuse of those existing (Gholami *et al.*, 2011).

A solution which makes it possible to define methodologies that fit specific necessities without losing the advantages coming from the exploitation of existing and experimented ones can be represented by the adoption of the ME paradigm (Ralyté, 2002; Cossentino *et al.*, 2006).

While the existing SOSE methods have proved their success in a specific task, they still have some weaknesses. Therefore, it is better to benefit from these advantages. To do that, it is essential to use method fragments, as this type of method offers to developers the possibility of selecting fragments of other methods and then assembling these different fragments to obtain a new SOSE method.

Indeed, there are many method fragments. To create our new SOSE method, we should choose a method fragment. In our case, we have chosen the open process framework (OPF) fragment, as this method fragment is the most complete. The OPF meta-model concerns with all the details to generate a method fragment, as it is composed of five main meta-classes. In addition, this meta-model contains already the OPF repository which holds a large number of components. Thus, it is easier to create a new SOSE method just by selecting from the repository the components which suit more with the situation of the developer. Also, this method fragment has been used before in the domain of SOSE method (Gholami *et al.*, 2011). So, it is not new to use it again and this can be seen as an advantage.

6. Related work

Most of the comparison frameworks have dealt with service identification and comparison frameworks of the existing methods. In Alani and Baghdadi (2012), the authors compared the methods with respect to the service identification approaches. Their survey aims at providing a method to identify and classify the services. Gu and Lago in Gu and Lago (2009) have 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. In another work (Gu and Lago, 2011), the same authors proposed a comparison framework that highlights aspects that are specific to SOA, aiming to differentiate the methodologies that are truly service-oriented from those that deal little with service aspects. As such, the authors suggest that the criteria defined in the framework can be used as a checklist for selecting an SOSE methodology. 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 combines the strengths of the examined approaches. Ramollari *et al.* (2007) presented a state-of-the-art survey of the some service-oriented engineering approaches and methodologies. They conclude that the service paradigm introduces unique requirements that should be addressed by innovative techniques. Baghdadi (2013) sketched out a framework to select an approach for WS and SOA, by comparing the generic methods.

From an ME perspective, we found the works of Gholami *et al.* (2010, 2011) to be close to ours. The aim of the authors in their first work (Gholami *et al.*, 2010) is to introduce a comprehensive evaluation framework for evaluating SOSE methodologies. This evaluation tool is appropriate for engineers to develop new methodologies, as well as project managers to select an appropriate methodology for a specific project. In fact, the framework is based on five aspects: development process, modeling language, service-oriented activities, service-oriented umbrella activities, and supportive features:

1. The development process perspective consists of sets of generic criteria defined and applicable for evaluating the development process part of any type of method. Having had an overall study of them, the authors leveraged and used these criteria in their proposed framework.
2. The modeling language is a set of criteria used for evaluating the modeling language part of the method.
3. The service-oriented activities consist of criteria that focus on the specific context activities (such as tasks, techniques or guidelines) that should be included in an appropriate SOSE-based development process. These criteria have been defined based on previous researches, SOSE literature, SOSE methodology challenges, SOA concepts and some good features of existing service-oriented methodologies. In fact, they have identified eight criteria that any SOSE method should respect. These criteria are:
   - business modeling;
   - SOAD;
   - service quality attributes;
   - service provisioning and consuming;
   - service testing, service versioning and evolution;
   - adaptable with legacy systems; and
   - cost estimation.
4. The service-oriented umbrella activities consider the more umbrella activities compared with common traditional methodologies.
5. The supportive features are a set of criteria around admirable features in SOSE methodology that are taken into consideration by methodology designers. The

considered criteria are architecture-based, service agility, process agility, maturity level and support tools.

Using these different criteria, the authors have analyzed the SOSE method proposed by Erl. They have used four descriptive degrees: "not addressed", "low", "medium" and "high", whereby they identified the strengths and the weaknesses of this method.

However, if we compare this framework with ours, we note that:

- We are interested in the ME approaches and techniques (if any) that are used by the existing SOSE methods. Whereas, in the framework of Gholami *et al.* (2010) this facet has not been taken into consideration.
- We have studied the different aggregates of methods used in the SOSE methods, which were not considered by Gholami *et al.* (2010); the authors have just been interested in the modeling language and the development process.
- The number of SOSE methods studied in our framework is greater than the number of SOSE methods studied in the framework of Gholami *et al.* (2010), who have limited their study to Erl's method.

In their second work (Gholami *et al.*, 2011), the authors have noted that the SOSE methods have specific characteristics. They have argued that these methods are different from the traditional methods. In addition, the authors have noted that the existing SOSE methods have not considered these different specifics. The authors also noted that given the variety of existing SOSE methods, it is hard for software engineers to decide which SOSE method best fits the specific needs of a project. Furthermore, specific tasks for SO in SOSE methods are tightly interwoven with traditional tasks, making it very hard for developers to extract and assemble the required specific tasks and activities related to SO. This asserts the evidence that there is no universal software development process that is appropriate for all situations. For that and to let developers to benefit from the different specific tasks designed to SO, the authors have decided to use the method fragments. These types of methods offer the opportunity to developers to select fragment from existing methods and put the fragments in a repository of reusable method fragments. Indeed, the authors have chosen the OPF method, as it has a repository of reusable method fragments (aka OPF); from which method engineers can select method fragments using suitable construction guidelines. To make this fragment method more suitable with the specifics of SO, the authors have enhanced the method base with new fragments of existing SOSE methods and then obtain a new SOSE method obtained by different fragments of other ones.

In conclusion, Gholami *et al.* (2011) have chosen to use a method fragment because there exist several SOSE methods that have some tasks and activities that have proved successful and it is a hard task for developers to extract these tasks and assemble them without using the method fragment.

Yet, their assertion is true only for some aspects (e.g. new features). Indeed, not only are there still some common characteristics with traditional methods, but also there still exist many SOSE methods that have proved successful.

Therefore, we argue that while we can take to profit of method fragments, the main difference between our work and the work of Gholami *et al.* appears in the cause and the need to use the method fragment.

Accordingly, this work aims at completing the existing frameworks toward a framework or ME techniques that consider the specifics of SO to engineer SOSE methods.

7. Conclusion and future work

The number of existing SOSE methods has given rise to many questions. Accordingly, many frameworks have been developed to compare the methods. Each of the frameworks has considered some. Yet, a few studies compared the methods with respect to ME techniques and the methods aggregates that are used to produce the methods. Therefore, this work aims at completing the existing frameworks toward a framework or ME techniques that take into account the specifics of SO to engineer SOSE methods.

In this work, we have proposed a comparison framework for SOSE methods. The framework uses four views: ME techniques, method aggregates, purpose and quality of the generated products (services). Each view has multiple facets.

The comparison has shown that while the existing SOSE methods have proved their success in a specific task, they still have some weaknesses. Therefore, it is better to benefit from the advantages of the existing ME techniques, notably method fragments, even if they need some alteration.

This work has practical implications as it:

- provides a better understanding of different views of SOSE methods; and
- assists the method engineers in deciding which ME technique is most suitable to their situation.

It also has theoretical implications, as the produced artifact that is the comparison framework opens new issues and provides a research roadmap toward SOSE ME.

This work has some limitations, specifically in terms of selected SOSE methods and also in terms of views.

This work could be further developed in many directions by exploring the open issues. For instance, the generation of a new ME technique and application of this new ME technique to the existing SOSE methods to see to what extent the existing methods may be situational.

References

Al-Rawahi, N. and Baghdadi, Y. (2005), “Approaches to identify and develop web services as instance of SOA architecture”, *Proceedings of the 2005 International Conference on Service Systems and Service Management*, *Beijing*, Vol. 1, pp. 579-584.

Alani, B. and Baghdadi, Y. (2012), “The application of service-orientation principles within service-oriented software engineering methods: a survey and comparison framework”, *Proceedings of the 2012 International Conference on Innovations in Information Technology*, *Al-Ain, UAE*, pp. 294-299.

Allen, P. (2007), “The service-oriented process”, available at: www.cbdiforum.com/reportsummary.php3?page=/secure/interact/20007-02/serviceorientedprocess.php&area=silve

Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S. and Holley, K. (2008), “Service-oriented modeling and architecture”, *IBM Systems Journal*, Vol. 47 No. 3, pp. 377-396.

Baghdadi, Y. (2006a), “Architecture for deploying e-business: business processes, web services-business interactions manager, and information systems”, *International Journal of Electronic Business*, Vol. 4 No. 1, pp. 18-39.

Baghdadi, Y. (2006b), "Reverse engineering relational databases to identify and specify web services with respect to service-oriented computing", *Information Systems Frontiers*, Vol. 8 No. 5, pp. 395-410.

Baghdadi, Y. (2012a), "A methodology for web services-based SOA realization", *International Journal of Business Information Systems*, Vol. 10 No. 3, pp. 264-297.

Baghdadi, Y. (2012b), "A survey on approaches to identify and develop Web-enabled services with respect to service-orientation and SOA: towards a value-oriented approach", *International Journal of Computer Applications in Technologies*, Vol. 45 No. 1, pp. 1-14.

Baghdadi, Y. (2013), "A comparison framework for service-oriented software engineering approaches: issues and solutions", *International Journal of Web Information Systems*, Vol. 9 No. 4, pp. 294-316.

Baghdadi, Y. (2015), "Service-oriented software engineering: a guidance framework for service engineering methods", *International Journal of Systems and Service-Oriented Engineering*, Vol. 5 No. 2, pp. 1-18.

Brinkkemper, S. (1996), "Method engineering: engineering of information systems development methods and tools", *Information and Software Technology*, Vol. 38 No. 4, pp. 275-280.

Catkinson, C., Bayer, J. and Muthig, D. (2000), "Component- based product line development: the KobrA approach", *Software Product Lines*, Kluwer, Vol. 576, pp. 289-309, ISBN: 0-7923-7940-3.

Cohen, S. (2007), "Ontology and taxonomy of services in a service-oriented architecture", *The Architecture Journal*, Vol. 11, No. 11 pp. 30-35.

Cossentino, M., Gaglio, S., Henderson-Sellers, B. and Seidita, V. (2006), "A meta-modeling-based approach for method fragment comparison", *Proceedings of The 11th International Workshop on Exploring Modeling Methods in Systems Analysis and Design*, *Luxembourg*.

Erl, T. (2005), *Service-Oriented Architecture: Concepts, Technology & Design*, Prentice Hall International, Boston, MA.

Erradi, A., Anand, S. and Kulkarni, N. (2006), "SOAF: an architectural framework for service definition and realization", *Proceedings of the SCC'06 IEEE International Conference on Services Computing*, *Chicago, IL*, pp. 151-158.

Gholami, M.F., Habibi, J., Shams, F. and Khoshnevis, S. (2010), "Criteria based evaluation framework for service-oriented methodologies", *Proceedings of the 12th International Conference on Computer Modelling and Simulation, Cambridge*, *IEEE Computer Society, Washington, DC*, pp. 122-130.

Gholami, M.F., Sharifi, M. and Jamshidi, P. (2011), "Enhancing the OPEN process framework with service-oriented method fragments", *Software & Systems Modeling*, Vol. 13 No. 1, pp. 361-390.

Gu, Q. and Lago, P. (2009), "Exploring service-oriented system engineering challenges: a systematic literature review", *Service Oriented Computing and Applications*, Vol. 3 No. 3, pp. 171-188.

Gu, Q. and Lago, P. (2010), "Service identification methods: a systematic literature review", *Towards a Service-Based Internet*, Springer, Vol. 6481, pp. 37-50, ISBN: 978-3-642-17693-7.

Gu, Q. and Lago, P. (2011), "Guiding the selection of service-oriented software engineering methodologies", *Service-Oriented Computing and Applications*, Vol. 5 No. 4, pp. 203-223.

Harmsen, A.F. (1997), *Situational Method Engineering*, Moret Ernst & Young Management Consultants, The Netherlands.

Inaganti, S. and Aravamudan, S. (2007), "SOA maturity model", available at: http://bpmg.orgwww.bptrends.com/publicationfiles/04-07-ARTThe%20SOA%20MaturityModel-Inagantifinal.pdf

Karlsson, F. and Gerfalk, P. (2004), "Method configuration: adapting to situational characteristics while creating reusable assets", *Information and Software Technology*, Vol. 46 No. 9, pp. 619-633.

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*, Vol. 2 No. 1, pp. 50-64.

Kontogogos, A. and Avgeriou, P. (2009), "An overview of software engineering approaches to service oriented architectures in various fields", *Proceedings of the 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, IEEE Computer Society*, pp. 254-259.

Lago, P. and Razavian, M. (2012), "A pragmatic approach for analysis and design of service inventories", *Service Oriented Computing – ICSOC 2011 Workshops*, Springer, Vol. 7221, pp. 44-53, ISBN: 978-3-642-31874-0.

Meier, F. (2006), "Service oriented architecture maturity models: a guide to SOA adoption", Master Dissertation, University of Skövde, Sweden.

Mittal, K. (2006), "Service Oriented Unified Process (SOUP)", available at: www.ibm.com/developerworks/library/ws-soa-method3/index.html#initial

O'Sullivan, J., Emond, D. and TerHofstede, A. (2002), "What's in a service?", *Distributed and Parallel Databases*, Vol. 12 No. 2, pp. 117-133.

Papazoglou, M.P., Andrikopoulos, V. and Benbernou, B. (2011), "Managing evolving services", *IEEE Computer Society*, Vol. 28 No. 3, pp. 49-55.

Papazoglou, M.P. and Van den Heuvel, W.J. (2006), "Service-oriented design and development methodology", *International Journal of Web Engineering and Technology*, Vol. 2 No. 4, pp. 412-442.

Pereira, C.M. and Sousa, P. (2004), "A method to define an enterprise architecture using the Zachman framework", *Proceedings of the 2004 ACM Symposium on Applied Computing, ACM Press*, *New York, NY*, pp. 1366-1371, ISBN: 1-58113-812-1.

Ralyté, J. (2001), "Vue stratégique sur l'ingénierie des méthodes", *INFORSID, Centre de Recherches en Informatique de l'université Paris 1*, Sorbonne, pp. 43-66.

Ralyté, J. (2002), "Requirements definition for the situational method engineering", *Proceedings of the IFIP WG8.1 Working Conference on Engineering Information Systems in the Internet Context*, *Kanazawa*, pp. 127-152.

Ralyté, J., Rolland, C. and Deneckère, R. (2004), "Towards a meta-tool for change centric method engineering: a typology of generic operators", *Advanced Information Systems Engineering*, Springer, Vol. 3084, pp. 202-218, ISBN: 3-540-22151-4.

Ramollari, E., Dranidis, D. and Simons, A.J.H. (2007), "A survey of service oriented development methodologies", *Proceedings of the 2nd European Young Researchers Workshop Service Oriented Computing*, *Leicester, UK*, pp. 75-80.

Rosane, S.H., Paulo, F.P., Flavia, C.D., Bruno, C., Everton, C. and Thais, B. (2014), "A systematic survey of service identification methods", *Service Oriented Computing and Applications*, Vol. 8 No. 3, pp. 199-219.

Sale, J.E.M., Lohfeld, L.H. and Brazil, K. (2002), "Revisiting the quantitative-qualitative debate: implications for mixed-methods research", *Quality and Quantity*, Vol. 36 No. 1, pp. 43-53.

Selmi, S., Kraiem, N. and Hajjami Ben Ghezala, H. (2005), "Toward a comprehension view of web engineering", *Web Engineering Lecture Notes in Computer Science*, Springer, Vol. 3579, pp. 19-29, ISBN: 3-540-27996-2.

Tilley, S., Gerdes, J., Hamilton, T., Huang, S., Müller, H., Smith, D. and Wong, K. (2004), "On the business value and technical challenges of adopting web services", *Journal of Software Maintenance and Evolution: Research and Practice*, Vol. 16 Nos 1/2, pp. 31-50.

Welke, R., Hirschheim, R. and Andrew Schwarz, A. (2011), "Service-oriented architecture maturity", *IEEE Computer Society*, Vol. 56 No. 1, pp. 61-67.

Wirsing, M., Hoelzl, M., Acciai, L., Banti, F., Clark, A. and Fantechi, A. (2008a), "SENSORIA patterns: augmenting service engineering with formal analysis, transformation and dynamicity", *Proceedings of the 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, *Berlin Heidelberg*, pp. 170-190.

Wirsing, M., Hoelzl, M., Koch, N., Mayer, P. and Schroeder, A. (2008b), "Service engineering: the SENSORIA model driven approach", *Proceedings of the Software Engineering Research, Management and Applications*, *Prague, Czech Republic*, pp. 14-16.

Zimmermann, O., Krogdahl, P. and Gee, C. (2004), "Elements of service oriented analysis and design, an interdisciplinary modeling approach for SOA projects", available at: www.ibm.com/developerworks/webservices/library/ws-soad1/lastaccessed2012

Further reading

Huhns, M.N. and Singh, M.P. (2005), "Service-oriented computing: key concepts and principles", *IEEE Internet Computing*, Vol. 9 No. 1, pp. 75-81.

About the authors

Boutheina Gherib is a PhD student in Software Engineering and Computer Science at National School of Computer Sciences (ENSI), Tunisia. Her research focuses on method engineering for service-oriented software engineering.

Professor Youcef Baghdadi received his PhD degree in computer science from the University of Toulouse 1 and his HDR in computer science from University Paris 1 Pantheon-Sorbonne, France. His research aims at bridging the gap between business and IT, namely, in the areas of cooperative information systems, IT, Web technologies, e-business, service-oriented computing, SOA and methods for service-oriented software engineering. He has published many articles in journals such as the *Information Systems Frontiers, Information Systems and E-Business, Electronic Business, Business Information Systems, Electronic Research and Applications, Web Grids and Services* and others. Youcef Baghdadi is the corresponding author and can be contacted at: ybaghdadi@squ.edu.om

Professor Naoufel Kraiem received his PhD from University of Paris VI and HDR from University of Paris1–Sorbonne–France. His research interests include IT adoption and usage, information modeling, software engineering, software product lines and CASE tools. His research work has been supported by funding of the CNRS, INRIA, MRT (Ministry of Research and Technology and Industry) and by the Commission of the European Communities under the ESPRIT Programmes (BUSINESS CLASS) and UNIDO (National Network of Industrial Information Project). He has had several articles published in many journals such as *Management Science Information Systems Research Communications of the ACM* and *IEEE Transactions*. He has been invited to present his research in many countries in North America, Europe, Africa and in the Middle East. Naoufel Kraiem has been member of over 20 program committees. He is in the board of AFCET and he is an old consultant at EDS–France from 1990 to 1996.

For instructions on how to order reprints of this article, please visit our website:
www.emeraldgrouppublishing.com/licensing/reprints.htm
Or contact us for further details: **permissions@emeraldinsight.com**