SOCA (2011) 5:203–223
DOI 10.1007/s11761-011-0080-0

ORIGINAL RESEARCH PAPER

Guiding the selection of service-oriented software engineering methodologies

Qing Gu · Patricia Lago

Received: 16 November 2010 / Revised: 16 February 2011 / Accepted: 18 February 2011 / Published online: 10 March 2011

Abstract Service-oriented computing is a paradigm for effectively delivering software services in a dynamic environment. Accordingly, many service-oriented software engineering (SOSE) methodologies have been proposed and practiced in both academia and industry. Some of these methodologies share common features (e.g. cover similar life-cycle phases) but are presented for different purposes, ranging from project management to system modernization, and from business analysis to technical solutions development. Given this diversity in the methodologies available in the literature, it is very hard for a company to decide which methodology would fit best for its specific needs. With this aim, we took a feature analysis approach and devised a framework for comparing the existing SOA methodologies. Different from existing comparison frameworks, ours specifically highlights aspects that are specific to SOA and aims to differentiate the methodologies that are truly service-oriented from those that deal little with service aspects. As such, the criteria defined in the framework can be used as a checklist for selecting a SOSE methodology.

Keywords Service-oriented software engineering · Evaluation framework · Service-Oriented software engineering methodology

1 Introduction

In today's global markets, modern enterprises need to respond quickly to business opportunities. To remain competitive, they are required to adjust their IT infrastructure to support their new business process models. Service-oriented architecture (SOA) is an architectural style that allows business functions to be exposed as reusable services across the network and enables business partners to search and discover those services on demand. A service-based application (SBA) built on SOA has the ability of discovering and composing services at runtime, if properly designed and implemented, to fulfill the ever changing business requirements. The agility, reusability and flexibility that SOA promises are the main drivers for a growing number of companies to adopt SOA as their architectural style.

Despite that the underlying philosophy behind SOA is stemmed from the principles of component-based design, the design and development of SBAs often require different development processes. One reason for these differences is that services are designed under open-world assumptions and physically owned by their service providers. Furthermore, without physically owning services, service consumers have to deal with issues (e.g. trust, security, and reliability) that become much more relevant in SOA development than it is in component-based development.

In addition to traditional software engineering (TSE) activities (such as coding, testing and deployment), developing SBAs requires identifying, discovering, and composing services. Hence, existing software development methodologies no longer fulfill the needs for developing SBAs. Systematic, disciplined and quantifiable approaches for designing, developing and maintaining SBAs are needed. We generalize these approaches as service-oriented software engineering (SOSE) methodologies. Different from some service-oriented approaches for specific SOSE activities, such as service identification methods, SOSE methodologies often provide guidance on multiple SOSE activities and aim at engineering SBAs rather than only designing atomic services or service compositions.

Q. Gu (✉) · P. Lago
Department of Computer Science, VU University,
Amsterdam, The Netherlands
e-mail: qgu@few.vu.nl

Although a mature and systematic software development methodology cannot guarantee high quality software, without such a methodology it would be even harder to trace the cause of errors when things go wrong. A number of SOSE methodologies have been proposed by vendors, such as SOAD [1], SOMA [2], and SOUP [3] by IBM, a methodology for service architectures by OASIS [4], and SO process by CBDI [5]. In the SOSE community, methods and processes for developing SBAs are becoming a hot research topic. SOSE methodologies proposed by Papazoglou et al. [6] and Chang [7] are examples of well known approaches from academia. Given these many methodologies, a common question for practitioners is how to select a methodology that suits their needs the best. More importantly, in SOSE community it is necessary to understand what a SOSE methodology entails to ensure the resulting SBAs being reusable and flexible.

With the objective of assisting the comparison and selection of SOSE methodologies, in this work, we have two research questions:

1. What are the characteristics of existing SOSE methodologies for engineering SBAs?
2. To what extent do they support service orientation?

The first question addresses the scope, features, attributes, or qualities of the SOSE methodologies in general. Its aim is to obtain a comprehensive overview of the existing approaches to engineering SBAs and investigate the distinctions and similarities among the SOSE methodologies. The answer to this question provides a preliminary evaluation of the methodologies, assessing their characteristics as software engineering approaches.

Complementary to the first question, the second question addresses specifically the service orientation aspects of the SOSE methodologies. Its objective is to analyze the way in which SOSE methodologies are different from the traditional ones and aspects that have been specifically designed to support service orientation. The answers to this question provide enterprises guidelines to select a methodology that best suits their needs, including adopting SOA as (part of) their IT portfolio and engineering SBAs that have the potential to achieve the benefits that SOA promises. Moreover, the answers would point out strengths and weaknesses of individual methodologies in supporting service orientation and thereby assist enterprises as well as vendors in enhancing their own SOSE methodologies.

To answer the questions above, we took a feature analysis approach and devised an evaluation framework for comparing the existing SOA methodologies. Feature analysis [8] is a qualitative evaluation method, offering a way to screen and assess a large number of software engineering methodologies by means of assessing their features (or characteristics). By conducting feature analysis, one can obtain an understanding of particular aspects of a methodology [9] and can systematically select or discard a methodology to achieve a particular goal [10].

Among the three alternative approaches (qualitative experiment, qualitative case study, and qualitative survey) to conduct a feature analysis [8], qualitative survey suits our work the best since it does not require the methodologies being used in practice or any experiment. As such, it is more feasible to study a large number of SOSE methodologies and gain insight into the common features they share and specific features provided by particular methodologies.

Differently from existing comparison frameworks (e.g. [11,12]), ours specifically highlights aspects that are peculiar to SOA and aims to differentiate the methodologies that are truly service-oriented from those that deal little with service aspects. As such, this framework highlights what a SOSE methodology should entail and can be used to compare and evaluate existing SOSE methodologies.

The remainder of this paper is structured as follows: Sect. 2 presents our service aspect-driven evaluation framework, including generic criteria and service-specific criteria; Sect. 3 presents the comparison of the existing SOSE methodologies using our evaluation framework and the selection guidelines; Sect. 4 discusses our observations on the needs for further improving the existing SOSE methodologies; Sect. 5 discusses some related work in the evaluation of SOSE methodologies. Section 6 provides our conclusions.

2 A service aspect-driven evaluation framework

To develop an evaluation framework for SOSE methodologies, we need to identify which features are of great importance to SOSE methodologies and relevant for comparison. For this purpose, we collected features from two perspectives, the generic and service-specific features.

The generic features are not specific to SOSE; they are properties common to software engineering methodologies in general, including SOSE methodologies. For instance, support project management and guidance in development activities and artifacts are features that are common to any software engineering methodologies. Instead, support in discriminating between service provision process and service consumption process is a service-specific feature.

In the following sections, we shall explain in detail the features that we consider relevant to SOSE methodologies and the criteria we derived from the features. Each criterion is associated with one evaluation question. The answer to the question can be used to compare and evaluate the SOSE methodologies.

2.1 Generic evaluation criteria

In their cataloging framework to evaluate software development methods, Karam and Casselman [13] discussed 21 properties, covering technical, usage, and managerial aspects. Some of these properties are product related, in the sense that they are properties of software systems, such as reusability, maintainability, and performance engineering; some other properties are process related, regarding the development process of software systems, such as life-cycle coverage, guidelines, degree of formality. In our work, we are interested in how SBAs should be engineered but not in how certain quality attributes are obtained. Therefore, we selected only the process-related properties as the generic features of SOSE methodologies.

Based on the selected properties, we derived a list of evaluation criteria given in Table 1: the first column presents the categories of properties defined in Karam and Casselman's framework, the second column presents the selected properties, the third column shows the criteria we derived from the properties. To ensure the criteria to be interpreted consistently, we also associated an evaluation question to each criterion. The answers to the evaluation question can be used to compare the SOSE methodologies. These answers can be of four types: narrative, YES/NO, scale, or multiple choices. A narrative answer is for open questions such as "what is the objective of the methodology"; a YES/NO answer can be given when a criterion is met or not; an answer with scale level can be given to a compound criterion where the degree of support provided by the methodology varies and can be judged on a scale (the reader may refer to Appendix A for an explanation of the judgment scale); when multiple choices for an answer are given, one may select one among them.

Table 1 Criteria for evaluating the generic aspects of SOSE methodologies

		Generic criterion	Evaluation question	Type of answer
Technical properties	Philosophy	GC1 Objective	What are the objectives of the methodology?	Narrative
	Life cycle coverage	GC2 Lifecycle	How many life cycle phases does this methodology cover?	Scale
	Work products & notations	GC3.a Artifacts	Does the methodology specify work products as results of specific activities?	Scale
		GC3.b Notations	Does the methodology specify the modeling of work products?	Scale
	Procedure	GC4 Procedure	Does the methodology describe the procedures of each covered life cycle phases?	Scale
	Guidelines, criteria, measures	GC5 Principles	Does the methodology provide guidelines or principles?	Scale
	Degree of formality	GC6 Formality	Are the technical aspects formal? i.e. being precise, unambiguous mathematical definition and can be reasoned about mathematically (logically)?	Scale
	Method specialization	GC7 Specialization	Does this methodology provide explicit support for tailoring to fit a particular organizations's needs or domain?	Yes/No
Usage properties	Automated support (tool support)	GC8 Tool	Are there tools available to support the methodology's techniques?	Yes/No
	Maturity/project history	GC9 Maturity	Has this methodology ever been applied in industry?	Yes/No
Managerial properties	Software development organization	GC10 Management	Does this methodology support project management?	Yes/No
	Ease of integration	GC11 Integration	What are the existing techniques that the methodology compatible with?	Narrative

2.2 Service-specific criteria

Service-specific criteria are specifically designed for evaluating the service-oriented aspects of the SOA methodologies. To derive these criteria, we used the seven differences between SOSE and TSE identified in our previous work [14] as service-specific features, motivated by the rationale that a SOSE methodology should support or provide guidelines to the aspects that are different from TSE. In the following, we shall discuss each of these differences and explain the criteria that we derived from them.

2.2.1 SOA is designed under open-world assumptions

Instead of assuming stable execution environment, SOSE must cope with high uncertainty in the external environment. Open-world assumptions postpone to runtime a number of design decisions that are usually made at design time. For instance, in component-based development a major emphasis is that the components of a software system can be bought and assembled at design time, whereas in service-oriented design a major emphasis is on services being dynamically discovered and composed in a SBA at runtime. As a result, a SOSE methodology should provide guidance to its users in what kind of uncertainties can be expected when engineering SBAs and how to handle such uncertainties (*Criterion SC1: Open-world assumption*).

2.2.2 Services are the building blocks

Instead of focusing on implementing a software system as a whole, SOSE focuses on composing coarse-grained discoverable services acting as building blocks of SBAs. This shifted focus should be explicitly supported by SOSE methodologies. To evaluate such support, a clear and concrete definition of services (*Criterion SC2: Service definition*) is essential for SOSE methodologies. Currently, a well-defined definition of services has not been commonly agreed. The definitions range from technical oriented to business oriented. A SOSE methodology, which guides the design and creation of services as well as their integration, should specify what a service means according to its own philosophy.

The second aspect relevant to services is the way that they are created (*Criterion SC3: The creation of services*). Depending on existing resources and goals of an organization, services can be realized in many different ways, ranging from greenfield development, identifying from existing software systems or business processes, and discovering from existing services that are possibly provided by third party-providers. Accordingly, the guidance that different organizations expect from SOSE methodologies would be different. For instance, an IT company that plans to develop some services and publish them for public consumption would require guidance on greenfield development; an enterprise that intends to migrate their existing legacy software assets to SBAs, instead, would require guidance on identifying services from both existing software and business processes. A SOSE methodology should make it clear how services as building blocks of SBAs are realized.

2.2.3 Additional development roles are involved in development

A software developer is not the only development role. This is rather split into three essential participants: service consumer, service provider, and service broker. Each participant is responsible for part of the service development tasks during the service life cycle; and collaborates as a whole to accomplish the development of SBAs. Depending on various development environments, development roles will have to be tailored. For instance, an organization that migrates existing systems to SOA requires different types of development roles from a service provider that aims at publishing reusable services to potentially unknown service consumers. The former requires a business process team for analyzing existing software and/or business assets and identifying services, while the latter requires a market scan team that understands the needs of the service consumption market.

Since one cannot assume the tasks and responsibilities of various roles required in the development of SBAs, a SOSE methodology should provide guidance on what development roles (*Criterion SC4: Development roles*) and their tasks (*Criterion SC5: Links between roles and activities*) are required when it is applied.

2.2.4 Services are open

In traditional software development, violation of the original architectural design of the system is extremely hard to handle as soon as the software systems are implemented, deployed and in execution, whereas in service-oriented development, the architecture of SBAs is often designed to be adapt to satisfy market changes, business demands, and customers' needs. Since well-designed SBAs are composed of autonomous and loosely coupled services, the independence between the services increases the ability of SBAs to dynamically take ever changing requirements into account. A SOSE methodology therefore should provide guidelines in how to design and implemented services and SBAs in such a way that they can evolve without hurdling their execution (*Criterion SC6: Architectural change*); moreover, a SOSE methodology should also describe what activities should be carried out at runtime in order to support dynamic evolution (*Criterion SC7: Runtime activities*).

2.2.5 Services are designed with multiple sets of (non)-functional requirements

A software application is engineered for a single set of (non)-functional requirements. In SOSE, instead, since the consumers as well as their needs are not completely known at design time (according to open-world assumptions), services are engineered with multiple sets of (non)- functional requirements, each fulfilling different groups of (potential) consumers with different quality requirements.

Non-functional requirements need to be considered over the entire service life cycle, not only because they play a crucial role as traditional software development, but also because they pertain to different service assets and hence have different scope. SOSE typically consists of two main processes, i.e. the development of services and the integration of services into a SBA. Different from TSE where non-functional requirements concern only software systems, in SOSE they concern individual services, service compositions and SBAs. Quality attributes of a single service are often hard to remain the same when it is composed into service compositions and consumed in SBAs, depending on the quality attributes of the other services and resources. As a SOSE methodology, it is important that it provides guidance on addressing non-functional requirements with different scope (*Criterion SC8: Non-functional requirements*).

Differently from a software application, a service might be consumed and shared among different consumers with different quality requirements depending on their application needs and business goals. To maximize reusability and manageability, a service is often realized with one implementation while provided with multiple SLAs to fulfill multiple quality requirements [15]. As a result, a service should be designed and developed in such a way that it can be consumed in different contexts. For instance, a GPS service consumed by a navigator application running on a mobile phone device would pose higher requirements in terms of response time as compared with the one consumed by a hotel finder application running on a server in an enterprise. This is because the former requires real-time service, while the latter requires off-line service. Accordingly, there is a strong need for SOSE methodologies to provide guidance on the engineering, provision, and consumption of services with multiple sets of (non)- functional requirements (*Criterion SC9: Variability*).

2.2.6 Services are consumed and executed remotely

Instead of buying and installing software locally to the own administrative domain, users of services (pay and) consume services that are executed remotely at the service provider's side (e.g. in the cloud).

This shifted ownership brings tremendous changes in SOSE. First of all, services can be completely isolated from their applications and can be published as software products over the network. The service provision process includes designing, developing, publishing, and maintaining services. More importantly, service are physically located and executed at the service provider's side. SOSE from the perspective of *service provision* means how to design and develop services that can be offered to a large number of consumers. Services can be provided either externally (e.g. to third parties), internally (e.g. own usage), or both.

Secondly, services are eventually composed and integrated into SBAs. After services are published by service providers, they need to be discovered, composed and integrated into SBAs to fulfill the specific needs of service consumers. SOSE from the perspective of service consumption means how to discover services and alternatives in case of service failure, how to compose the discovered services or in-house developed services to achieve certain reusable business functions and how to integrate services or service compositions with existing software resources.

Thirdly, when they are published, services need to be cataloged in a service registry or repository to ensure discoverability and reusability from both known and potential consumers. SOSE from the perspective of *service brokerage* means create and maintain well-functional service registries, and keep the interfaces of registered services up-to-date.

In practice, an enterprise that needs to perform some SOSE activities may have different types of business. Taking Amazon as an example:

- Amazon provides messaging services (e.g. Amazon Simple Queue Service (SQS)) or payment and billing services (e.g. Amazon Flexible Payments Service (FPS)) to its business partners. To this purpose, it acts as a service provider performing the process of service provision.
- Amazon composes existing services to achieve a specific business goal and provides the resulting composite services (e.g. delivery services based on shipping companies) to its business partners. To this purpose, Amazon still acts as a service provider, but in this case it performs both service consumption (i.e. discover and select services provided by shipping companies) and service provision (i.e. construct composite services and publish them).
- Amazon maintains a cloud of web services from which one may select the needed services to realize its own e-business functions. To this purpose, Amazon acts as a service broker performing the process of service brokerage.
- Moreover, Amazon may also act as an application builder when it integrates all the services needed to a web application that can be directly used by the end users. In this case, Amazon needs to perform the process of service consumption and/or service provision and/or service brokerage.

Because of the shift in ownership, SOSE should cover different types of process from the perspectives of a service provider, service consumer or system builder. Accordingly, it is of great importance to understand the target perspective (*Criterion SC10: Perspective*) that a SOSE methodology should support.

2.2.7 Services are cross-organizational

Multiple roles in developing, maintaining, executing, and evolving SBAs are not completely independent but rather requiring collaboration. For instance, when a service provider publishes a service through a service broker, they need to collaborate to keep the service up-to-date whenever the service description is updated. Since the development of SBAs requires more development roles as compared to traditional software systems [16,17], the interaction between these roles becomes more complex, too.

Moreover, since services are not executed locally at the consumer's side, control of services is often highly distributed and crosses trust and organizational boundaries. The interaction and collaboration between multiple development roles therefore cross the boundaries of the organizations as well. As a result, a SOSE methodology should provide guidance on the interactions between the development roles that are potentially distributed at multiple organizations (*SC11: Multiple organizations*).

In summary, we identified 11 evaluation criteria that are specifically relevant to SOSE methodologies. An overview of these service-specific criteria are give in Table 2, each summarizing question to be answered when evaluating a SOSE methodology and expected type of answer.

Table 2 summarizes the 11 service-specific evaluation criteria we identified, the corresponding evaluation questions (to be answered in evaluating a SOSE methodology) and the expected type of answer.

3 Comparison of existing SOSE methodologies using the evaluation framework

3.1 Overview of the selected SOSE methodologies

The aim of this work is to understand what SOSE methodologies entail rather than gaining insight into the state of the art. To support our aim we selected a mix of 12 prominent SOSE methodologies discussed and compared in already existing evaluation frameworks (discussed in detail in Sect. 5). The selected methodologies have been proposed from both industry and academia and published in both journals (e.g. [6]), conferences (e.g. [18]), and white papers (e.g. [2]).

To show the usability of the evaluation framework, we compared the selected SOSE methodologies based on the criteria presented in Sect. 2. In the following, we shall briefly introduce the 12 SOSE methodologies we selected for evaluation.

The *CBDI-SAE* [5] methodology is part of the CBDI-SAE SOA reference framework (RF) published by the CBDI forum. The process consists of four key discipline areas, including consume, provide, manage, and enable. To tailor the process to the needs of different audiences, multiple

Table 2 Criteria for evaluating the service-specific aspects of SOA methodologies

Service-specific criterion	Evaluation question	Type of answer
SC1 Open-world assumption	Does the methodology support open-world assumptions?	Scale
SC2 Service definition	Does the methodology give a definition of services?	Yes/No
SC3 The creation of services	Does the methodology describe how services are created?	Narrative
SC4 Development roles	Does the methodology explain roles?	Scale
SC5 Association of roles to activities	Are the activities associated with the roles?	Scale
SC6 Architectural change	Does the methodology support architectural change after services are deployed and in execution?	Scale
SC7 Runtime activities	Does the methodology support runtime activities?	Scale
SC8 Non-functional requirements	Does the methodology consider (non)-functional requirements?	Scale
SC9 Variability	Does the methodology support variability among different sets of service consumers and different contexts	Scale
SC10 Perspective	From which perspective is the methodology described from?	Service consumption, service provision, service brokerage
SC11 Multiple organizations	Does the methodology support SOA development by multiple organizations in collaboration?	Scale

process views (such as flow view, existing organizational view, life-cycle stages view) can be created to assist the engineering of SBAs from different perspectives.

Chang [7] proposes a methodology based on SOAD, consisting of six phases. The results of each phase refer to one or more key artifacts specified in SOAD. Chang's methodology specifically addresses concerns relating to dynamically adaptable services. To deal with service variability and mismatch, Chang's approach considers three types of variation points in service design (workflow, service composition, and logic) and three types of service mismatch (interface, functional, and nonfunctional).

Steve Jones from (*OASIS*) [4] developed a methodology for service architectures, providing mechanisms for planning, managing, and delivering projects using SOA techniques. The methodology follows a broadly four step process answering four questions (what, who, why, and how). The aim of this method is to describe how a service architecture can be defined, rather than how it can be delivered.

The Web services development life-cycle methodology (*SLDC*) [6] utilizes an iterative and incremental process based on several well-established process models from TSE, such as the rational unified process (RUP), component-based development (CBD) and business process modeling (BPM). The methodology consists of eight phases that cover the whole service life cycle, ranging from planning to deploying, monitoring and managing SBAs.

The *SeCSE* [19] methodology is the main process adopted by the SeCSE[1] project (a European Union-funded project) in the realm of service-centric systems (SCS) engineering. At the highest level the SeCSE methodology is represented by three important functional areas: *service engineering functional area* where the service developer develops services; *service acquisition/provisioning functional area* where service provider delivers the services in the marketplace; and *service-centric system engineering functional area* where the consumers can build and manage SBAs based on their choices about which services suit their needs. Designed with service adaptation in mind, the SeCSE methodology also supports runtime service composition and recovery management.

SOAD [1] by IBM is a methodology based on existing modeling disciplines such as object-oriented analysis and design (OOAD), enterprise architecture (EA) frameworks, and BPM. It proposes elements, such as domain decomposition, service categorization and aggregation, and semantic brokering, that need to be considered in service-oriented analysis and design.

Service-oriented architecture framework (*SOAF*) [20] is an academic methodology presenting an architecture-centric framework. Its goal is to ease the definition, design, and realization of SOA to achieve a better business and IT alignment. To this end, the methodology uses two types of business process models: *To-be modeling*: a top-down business-oriented approach describing the candidate solution, and *As-is modeling*: a bottom-up approach describing current business processes and the problem space.

Service-Oriented Development In a Unified fraMework (*SODIUM*[2]) [21] is an academic project involving international research and industrial partners. The methodology developed in this project focuses on how to define new services based on compositions of reusable coarse-grained services. To this end, SODIUM proposes a set of models, languages, middleware, and tools to be adopted for engineering SBAs.

SOMA [2] is an iterative and incremental methodology developed by IBM, aiming at the identification, modeling, and design of services and SBAs. The SOMA methodology consists of seven phases starting from business modeling and transformation to solution management. Using the modeling tools, SOMA breaks down the business process into a component view. The solution is modeled based on SOA reference model, which defines a layered systems architecture.

The *SOSE* [18] framework is a methodology aiming at developing methods and tools to improve quality and profitability of SBA development. It suggests that service and component design should always start by creating a business case to justify the project implementation. Combining component-based and service-oriented development, the framework defines three levels of granularity: system-level component (SLC), business service component (BSC), and component level.

The service-oriented unified process (*SOUP*) [3] was a methodology proposed by Knual Mittal from IBM, using the best elements from RUP and XP. Aiming at assisting the establishment and management of SOA projects, SOUP has been specifically designed for both initial SOA deployment and ongoing SOA management.

A methodology for engineering a *TRUE* SOA that allows companies to flexibly adapt to changing market demands was proposed by Engels et al. [22]. In this methodology, the business of an enterprise is organized in a service-oriented way (as a set of business services) and the enterprise IT architecture is structured according to those business services.

3.2 Comparing the generic aspects of the SOSE methodologies

Using the generic evaluation criteria described in Sect. 2.1, we evaluated the 12 SOSE methodologies in terms of the development process. The results of the evaluation are presented in Table 3. In the following, we shall discuss the results

[1] http://www.secse-project.eu.

[2] http://www.atc.gr/sodium.

Table 3 Comparing the generic aspects of the SOSE methodologies

Generic criterion	CBDI-SAE	SDLC	SeCSE	SOMA	OASIS	SOAD	SOSE	TRUE	CHANG	SODIUM	SOAF	SOUP
GC1 Objective	Engineering process	Engineering process	Engineering process	Engineering process	A&D	A&D	A&D	A&D	Service composition	Service composition	SOA migration	Project management
GC2 Lifecycle	See Table 4											
GC3.a Artifacts	+++	+	+++	++	+++	–	++	+++	+++	+++	+++	+++
GC3.b Notations	–	++	++	+++	++	–	++	+++	+++	++	+	–
GC4 Procedure	++	++	+++	++	+++	–	++	+++	++	++	+++	++
GC5 Principle	+++	+++	–	+++	++	–	++	++	++	++	+++	–
GC6 Formality	–	–	–	–	–	–	–	–	–	–	–	–
GC7 Specialization	No	No	No	No	No	No	No	No	No	No	No	No
GC8 Tool	–	++	+++	–	–	–	–	–	–	+++	–	–
GC9 Maturity	NA	No	Yes	Yes	Yes	Yes	Yes	Yes	NA	No	Yes	NA
GC10 Management	++	–	++	++	+++	–	–	–	–	–	++	+++
GC11 Integration	–	RUP, CBD, BPM	–	OOAD, RUP, BPM	–	OOAD, BPM	–	–	SOUP, SOMA	–	–	RUP, XP

in detail and explain how each criterion assists the selection of SOSE methodologies.

3.2.1 GC1 Objective

Despite that the ultimate goal of SOSE methodologies is to deliver SBAs that meet the requirements of their users, the objective of individual SOSE methodologies often varies. Some of the SOSE methodologies are designed to provide guidance to the entire engineering process, such as CBDI-SAE, SDLC, SeCSE, and SOMA; some focus only on the analysis and design of the architecture of SBAs, such as OASIS, SOAD, SOSE, and TRUE; some concern specific engineering issues, e.g. Chang and SODIUM focus on dynamic service composition, SOAF focuses on SOA migration; and SOUP aims at assisting project management.

Selection guideline
Knowing the objective of the SOSE methodologies is essential for enterprises to select the one that suits their needs at best. This criterion therefore can be used as the first step to filter out relevant candidates for selection. If for instance an enterprise aims at composing existing services for internal or external uses, it may consider Chang and SODIUM as candidate methodologies.

3.2.2 GC2 Lifecycle

To compare the lifecycle coverage of the SOSE methodologies, we used the service life cycle model [23] proposed in our previous work. The description of each phase of the life cycle model is given in Appendix B. Based on this model, we analyzed each SOSE methodology reported in Table 4 the coverage of each phase using the scale defined in Appendix A. It is reasonable that the SOSE methodologies aiming at providing guidance for the entire engineering process (e.g. SDLC and SeCSE) cover the largest number of life cycle phases while the methodologies that focus on the analysis and design of an architecture (e.g. SOSE and TRUE) only cover few life cycle phases that are related to requirements engineering and service design.

When looking at the individual life cycle phases, we can see that business modeling and service design are nearly covered by all the SOSE methodologies, while application-specific phases such as application design and implementation are not covered by any of the SOSE methodologies. The main reason for leaving out application-specific phases is that all these SOSE methodologies focus on service-specific phases including the design, development and delivery of services (and service compositions) while regarding application-specific phases as part of traditional software engineering process (which is out of the scope of these SOSE methodologies).

Table 4 Comparing the life cycle coverage of the SOSE methodologies

Role	Life cycle activity	CBDI-SAE	SDLC	SeCSE	SOMA	OASIS	SOAD	SOSE	TRUE	CHANG	SODIUM	SOAF	SOUP
Service provider	Market scan	–	++	–	–	+	–	–	–	–	–	–	–
	Requirements engineering	++	+++	–	++	++	++	+++	++	–	–	++	++
	Business modeling	++	+++	+++	+++	++	++	+++	++	++	–	+++	–
	Service design	++	+++	+++	+++	++	+++	+++	+++	++	–	+++	++
	Service development	++	++	++	+	–	–	–	–	++	–	+++	++
	Service testing	++	+++	+++	+	–	–	–	–	–	–	–	–
	Service publishing	++	+++	+++	+	–	–	–	–	–	++	–	++
	Service provision	++	+++	+++	+	–	–	–	–	–	++	–	–
	Service monitoring	–	+++	+++	+	–	–	–	–	–	–	–	–
	Service management	++	+++	+++	+	–	–	–	–	–	–	–	–
Service broker	Registry selection	+	–	–	–	–	+	–	–	–	–	–	–
	Registry update	+	–	–	–	–	–	–	–	–	+++	–	–
	Registry maintenance	+	–	–	–	–	–	–	–	–	–	–	–
Service consumer	Service discovery	–	+	+++	+	–	–	–	–	++	+++	–	–
	Service composition	+	+	+++	+	–	–	–	–	++	+++	–	–
	Service negotiation	–	–	+++	–	–	–	–	–	–	–	–	–
	Service invocation	–	++	+++	++	++	–	–	–	–	–	–	–
	Service monitoring	–	+++	+++	+++	–	–	–	–	–	–	–	–
Application builder	Requirements engineering	++	–	+++	–	–	–	–	–	++	+++	–	–
	Application design	–	–	–	–	–	–	–	–	–	–	–	–
	Application implementation	–	–	–	–	–	–	–	–	–	–	–	–
	Module testing	–	–	–	–	–	–	–	–	–	–	–	–
	Application testing	–	–	–	–	–	–	–	–	–	–	–	–
	Application maintenance	–	–	–	–	–	–	–	–	–	–	–	–

Moreover, registry-related phases are only marginally covered by three SOSE methodologies (CBDI-SAE, SOAD and SODIUM) and no detailed guidance is given. Surprisingly, phases that are specifically important to SOSE such as service monitoring, service discovery and service composition are only covered by few methodologies.

According to the life cycle model, requirements engineering is often performed by two roles: a service provider and an application builder. Some SOSE methodologies (e.g. SDLC) support the collection of requirements from the market: in this case the role played in using the methodology is that of a service provider that delivers services to its *external* clients. Some other methodologies (e.g. SODIUM) support the collection of requirements from its internal users: in this case the role played in using the methodology is that of a service provider, too, but services are delivered to *internal* clients instead. Some SOSE methodologies (e.g. SeCSE) support the collection of system requirements from its end users: in this case the role played is that of an application builder.

Selection guideline:
Some enterprises look for guidance for specific activities rather than fully fledged methodologies. In this case, knowing the phases covered by the SOSE methodologies is crucial for selection. An enterprise that aims at building SBAs, for instance, should select a SOSE methodology and apply it with a traditional software engineering methodology since obviously the analyzed SOSE methodologies do not provide any support for application-related activities.

3.2.3 GC3.a Artifacts

Most of the methodologies provide detailed description of the outcomes of each engineering activity. Only SOAD does not give any information about its produced artifacts.

Selection guideline:
If the goal of an enterprise is specifically to produce certain outputs, this criterion should be used to drive selection, and obviously SOAD would not be a candidate for such goal.

3.2.4 GC3.b Notations

Most of the methodologies that specify their artifacts also provide some guidance on the modeling techniques, except

for CBDI-SAI, SOAF, and SOUP. Chang's approach explicitly indicates when traditional modeling techniques (e.g. commonality and variability modeling) can be used and when service-specific modeling techniques (e.g. service decision model) are needed. Some of the methodologies not only specify the modeling of their artifacts but also give some examples (e.g. service component specification and goal-service model in SOMA, UML domain model in TRUE, business process model in SDLC). Many other methodologies only name some models (e.g. business process map in OASIS, service interaction model in SeCSE, abstract composition model in SODIUM) that need to be created but do not provide further guidance on how to create them.

Selection guideline:
As enterprises have tools (and their supported notations) already in place, this criterion is relevant to select methodologies that could be potentially supported by the existing tools, and (if not) to explicitly assess necessary investments. Though, excluding TRUE that uses UML, all other methodologies only use types of models, which could be then checked against supporting notations.

3.2.5 GC4 Procedure

Most of the methodologies describe the procedures (i.e. a specified series of actions or operations which have to be executed) of each covered life cycle phase, except for SOAD. As indicated by the scale, some methodologies (e.g. OASIS, SeCSE, SOAF) provide very detailed guidance, describing comprehensively how to carry out each engineering activity and its sub-activities; while many other methodologies only explain shortly what the engineering activities (and their sub-activities) are and what they entail, but no step-by-step guidance is given.

Selection guideline:
This criterion is extremely useful for enterprises that do not have experience in developing SBAs and need (procedural) guidance on the way each life cycle phase should be carried out.

3.2.6 GC5 Principles

Most of the methodologies discuss guidelines or principles for the development process they support. However, only CBDI-SAE, SDLC, SOAF, and SOMA explain in detail the design principles they used, among which separation of concerns, loose coupling and business-IT alignment are the principles commonly considered. SOAD and SOUP only briefly mention the engineering activities they support but do not enter the details.

Selection guideline:
Similar to criterion *GC4 Procedure*, this criterion is useful for enterprises that lack experience and need guidance on design principles.

3.2.7 GC6 Formality

None of the methodologies support formal reasoning.

Selection guideline:
If an enterprise is seeking for SOSE methodologies that use mathematical definitions and can be mathematically reasoned about, obviously none of the analyzed methodologies is suitable.

3.2.8 GC7 Specialization

All analyzed methodologies are domain-independent and they can be used in any organizations. However, none of the methodologies provides explicit support for how to tailor them to fit a particular organization's needs or domain.

Selection guideline:
Enterprises that plan to apply SOA in specific business domains should check for SOSE methodologies yielding any domain-specific features (n/a in the SOSE methodologies we analyzed so far).

3.2.9 GC8 Tool

Only three methodologies are (partially) tool supported. Thanks to their industrial applications, the two methodologies developed within European projects, SeCSE and SODIUM, are very well supported by the tools specifically developed for them. In SeCSE, using tools to support engineering activities and to produce work products is well explained. In SODIUM, three tools (the composition studio, the SODIUM runtime environment, and the SODIUM composition repository) are introduced and their use in engineering activities, such as modeling, service discovery, service selection, web service wrapper generation and service deployment. Different from these two industrial methodologies, SDLC does not develop its own tools; instead, it recommends some existing tools available in the market to support some engineering activities. For instance, when building business architecture SDLC suggests using IBM Rational Portfolio Manager to gain insight into the business benefits and costs of the SOA services portfolio; and for design and developing business processes SDLC suggests to use some automated tools such as IBM's WebSphere Business Modeller.

Selection guideline:
This criterion is crucial for selection if an enterprise is seeking for methodologies that are tool supported. From our analysis the SOSE methodologies that are tool supported are from industry and consequently these tools are not publicly available. Only SDLC introduces some tools that are available on the market.

3.2.10 GC9 Maturity

The maturity of the SOSE methodologies relies on their application in real-life projects. Only when applied, the advantages and shortcomings of the SOSE methodologies become apparent and therefore can be further evolved to improve its maturity. Thanks to their industrial collaboration, most of the SOSE methodologies proposed in industry have already been applied, such as OASIS, SeCSE, SOAD, and SOMA. Only CBDI-SAE does not report any real-life case studies. Among the methodologies proposed in academy, SOAF and SOSE have been applied in securities trading and electricity market domain, respectively. TRUE has been applied and validated in many large-scale industrial projects. The other methodologies do not provide information about their application.

Selection guideline:
Some enterprises are open to try newly developed SOSE methodologies, testing their strengths or weaknesses and distilling new ideas or techniques from them. Some other enterprises are not flexible to experiment with immature methodologies due to the risks intrinsic of their business domain (e.g. banking domain), and require tested and validated ones. Knowing the level of maturity of the methodologies is useful for selection.

3.2.11 GC10 Management

SOA project management refers to a business activity (often performed by a project manager) that plans, starts, controls and ends the engineering of a SBA to achieve a particular aim. It is of great importance to any SBA development when putting into industrial practices. OASIS and SOUP are designed specifically from the perspective of project management. In OASIS, deliverables required in each stage of the development process are defined in terms of their description and expected duration, the required team members are suggested and an example of the project plan is given; in SOUP, not only key deliverables are defined but also the development process for an initial project (when an SBA has not been built) is differentiated from an ongoing project (when an SBA has been built but requires maintenance). Other methodologies, although not designed for project management, consider project management as part of their engineering process. For instance, in CBDI-SAE two terms (project and project profile) that are dedicated to project management are embraced by the CBDI-SAI SOA reference framework; in SeCSE the actions on the working products such as read-only, create or modify are defined, which assists the control of the working products; SOAF supports project management in the sense that it suggests a list of key deliverables for planning the projects but their descriptions are missing; in SOMA, initiating project management activities are suggested in the phase of solution management but their descriptions are also missing. The remaining methodologies do not include any guidance for project management.

Selection guideline:
This criterion is extremely useful for enterprises that lack experience in SOA project management. From our analysis the supported management activities are: guidance for project deliverables, team setup, project planning, setup of greenfield projects, and managing pre-existing projects.

3.2.12 GC11 Integration

The ease of integration of a SOSE methodology refers to its compatibility with existing techniques. Among the studied SOSE methodologies, OOAD, CBD, BPM, IBM RUP, XP, and BPM are techniques from TSE. OOAD, an approach that models systems as a group of interacting objects, is used by SOAD and SOMA; CBD, built upon OOAD principles and focused on the separation of concerns of software systems, is explicitly used by SDLC; RUP, an iterative software development process framework created by the Rational Software Corporation, is used by SDLC, SOMA, and SOUP; XP, a type of agile software development that aims at improve the productivity of software by adopting customer requirements or feedback in frequent software releases, is used by SOUP; BPM, a management approach aimed at aligning different aspects of an organization with the requirements of its clients, is used by SDLC, SOMA and SOAD. Chang is the only one that is built upon existing SOSE methodologies, including SOUP and SOMA.

Selection guideline:
Some enterprises have already been working with existing techniques. For these enterprises, it would be easier to apply a SOSE methodology that is compatible with the familiar techniques. This criterion is useful for enterprises to be aware of the methodologies that allow them to reuse their own integration knowledge and experience.

3.3 Comparing the service-specific aspects of the SOSE methodologies

Using the service-specific criteria described in Sect. 2.2, we evaluated the 12 SOSE methodologies from the perspective of service orientation. The results of the evaluation are presented in Table 5 (sorted by their objectives). In the following, we shall discuss the results in detail and explain how each criterion assists the selection of SOSE methodologies.

3.3.1 SC1 Open-world assumption

This criterion is used to evaluate whether a SOSE methodology provides guidance in handling the uncertainty (e.g. unforeseen service consumers, execution context and service usage) during the development of SBAs. Based on our evaluation, there are only two methodologies (SOMA and SDLC) embrace open-world assumptions. SOMA is composed of capability patterns, executed in all phases with different degrees of elaboration and precision. For instance, exposure decisions are first made based on the information about the services known at the identification phase and these decisions are then elaborated and refined in the specification phase when more information about the non-functional requirements of the services are known. To give another example, SOMA supports the recombination of services in unanticipated service context by developing a service context diagram depicting the service ecosystem. In such a service ecosystem, a group of related services as well as their providers and consumers, input and output, and underlying systems that implement them are all illustrated.

SDLC emphasizes the importance of different types of coupling in service design to address uncertainties. For instance, it suggests that business processes should not depend on specific representational or implementation details (representational coupling); connection channels between services should be unaware of who is providing the service (identity coupling); and a sender of a message should rely only on those effects necessary to achieve effective communication (communication protocol coupling). In the rest of the SOSE methodologies, although loose coupling is often considered in service design, they do not explicitly consider any open-world assumptions.

> Selection guideline:
> If an enterprise faces many uncertainties during the development of SBAs, this criterion is important for selecting a methodology that is capable of dealing with these uncertainties. From our analysis uncertainties of non-functional requirements, unanticipated service context, unknown implementation details, unknown service providers, and unknown communication protocols are currently addressed by some of the methodologies.

3.3.2 SC2 Service definition

Surprisingly, only half of the SOSE methodologies provide a definition of services. Some methodologies define services from a technical perspective. For instance, SOAD defines services as "logical groupings of operations"; SOSE defines services as "components with published interfaces."; and in SOMA, services are "first-class constructs of service orientation". OASIS and TRUE, instead, define services from a business perspective. In OASIS, services represent "what the business does" and in TRUE, a service is the output of a service provider toward a service consumer. Among all the definitions, Chang provides the most comprehensive one, which defines services from both service consumer ("a unit of functionality with a certain service-level agreement expected by consumers") and service provider ("a unit of deploying functionality which is publishable in WSDL standard") point of view.

> Selection guideline:
> Each enterprise has its own understanding of what services entail. To ensure the resulting services or SBAs do fulfill the goals of enterprises, being aware of how services are defined in a SOSE methodology is crucial for them to select the right ones.

3.3.3 SC3 The creation of services

Services as building blocks of SBAs are designed and developed differently from methodology to methodology. In general, the methodologies address the creation of services in terms of two tasks: service identification and service realization. Except for CBDI-SAE and SeCSE that do not suggest any service identification methods, all the other methodologies provide guidelines in identifying (abstract, not implemented) services from different resources, such as requirements from service consumers and stakeholders (e.g. Chang, SOUP), enterprise knowledge (e.g. OASIS and TRUE), business domain (e.g. SDLC, SOAF, SODIUM, SOSE), legacy systems (e.g. SOAD), or a mix of these resources (e.g. SOMA).

In terms of service realization, many SOSE methodologies (CBDI-SAE, SDLC, SOAF, SOMA, SOUP) support transforming legacy system into services by either wrapping existing components or re-engineering and re-factoring legacy applications. Interestingly, the methodologies that support legacy transmission often suggest greenfield development as well, by developing new services from scratch. This is quite understandable in that migrating legacy system does

Table 5 Comparing the service-specific aspects of the SOSE methodologies

Service-specific criterion	CBDI-SAE	SDLC	SeCSE	SOMA	OASIS	SOAD	SOSE	TRUE	CHANG	SODIUM	SOAF	SOUP
SC1 Open-world assumption	–	+++	–	+++	–	–	–	–	–	–	–	–
SC2 Service definition	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
SC3 The creation of services	No service identification; legacy transition and greenfield development	Identified by decomposing business process; created by discovery, legacy transition and greenfield development	No service identification; created by discovery and composition	Identified from organizational resources, domain decomposition, and existing asset analysis; created by legacy transition and greenfield development	Identified by analyzing the broad "what" of the enterprise; no realization	Identified from existing legacy systems, business processes and rules; no realization	Identified by analyzing business domain; no realization	Identified from enterprise architecture; no realization	Identified from requirements; created by discovery and composition	Identified by decomposing business processes; created by discovery and composition	Identified from business decomposition and existing IT portfolio; created by legacy transition and greenfield development	Identified from requirements; created by legacy transition and greenfield development
SC4 Development roles	–	–	+++	–	++	–	++	–	–	–	–	–
SC5 Association of roles to activities	–	–	+++	–	–	–	+++	–	–	–	–	–
SC6 Architectural change	–	++	+++	++	–	–	–	++	+	–	++	–
SC7 Runtime activities	–	++	+++	++	–	–	–	–	++	+++	–	–

Table 5 continued

Service-specific criterion	CBDI-SAE	SDLC	SeCSE	SOMA	OASIS	SOAD	SOSE	TRUE	CHANG	SODIUM	SOAF	SOUP
SC8 Non−functional requirements	−	+++	+++	++	−	++	++	++	++	+++	++	++
SC9 Variability	−	+++	+++	−	−	−	−	−	+++	−	−	−
SC10 Perspective	Service provision, service consumption	Service provision	Service provision, service consumption	Service provision	Service provision	Service provision	Service provision	Service provision	Service consumption	Service consumption	Service provision	Service provision
SC11 Multiple organizations	++	+	+++	−	‡	−	−	−	−	++	+++	−

not address new business requirements and hence services with new business functions have to be developed.

SeCSE, SODIUM, and Chang support the development of SBAs through composing services that are discovered from existing services. In SeCSE, services are discovered in three phases: in requirements definition phase services are discovered as candidate services from the service registry if they fit the business model and system requirements, in design time services are further discovered from candidate services if they are compatible with specific system architectures and workflows, and in runtime SeCSE supports the discovery of alternatives for replacing services that become unavailable or fail to meet certain requirements. In SODIUM, heterogeneous services (i.e. web, p2p and grid services) are discovered from heterogeneous registries and networks with the use of the USQL Engine if the semantic descriptions and QoS requirements are met. In Chang, published services that correspond to the requirements are discovered.

OASIS, SOAD, SOSE, and TRUE specifically focus on the design and analysis part of the SOSE process, providing guidance only on service identification but not on service realization.

> Selection guideline:
> Some enterprises may have already had a clear vision on the services needed for reaching their business goals or have already applied some methods to identify services. Some other enterprises do need support of identifying services from their existing resources. In either case, knowing the different levels of support of service creation provided by SOSE methodologies facilitates the selection of SOSE methodologies.

3.3.4 SC4 Development roles

Despite of the importance of development roles required during any software development process, many SOSE methodologies do not provide any guidelines on these roles. Only SeCSE, OASIS, and SOSE explicitly explain development roles or actors. In SeCSE, for each engineering activity, a list of actors as well as their responsibilities is described. As such, SeCSE provides a complete list of development roles required when it is applied in practice. OASIS does not provide a complete list of the development roles; instead, it only lists the roles required in the planning phase of the project, in which the key stakeholders who are required to create the service architecture should be identified. In SOSE, the development process consists of two parts, business case and service & component design. Only in the business case, the required roles and their responsibilities are described.

> Selection guideline:
> Enterprises, especially the ones that have little experience with developing SBAs, often do not know what

kind of expertise and personnel are required before a SOA project starts. This criterion is especially useful for enterprises when they have such doubts. In addition, enterprises may select the ones that fit their needs based on the existing human resources to minimize the need for recruitment.

3.3.5 SC5 Association of roles to activities

Among the three SOSE methodologies (SeCSE, OASIS, and SOSE) that do explicitly explain the development roles, OASIS does not associate the development roles to any engineering activities. Rather, OASIS only briefly lists some key roles required in preparing and planning a SOA project. In both SeCSE and SOSE, the development roles are explicitly associated with their engineering activities.

Selection guideline:
Similar to criterion *SC4 Development roles*, this criterion is useful for enterprises that have little experience with developing SBAs to select SOSE methodologies that clearly explain responsibilities of required team members.

3.3.6 SC6 Architectural change

Despite that adaptability is of great importance to SBAs, about half of the SOSE methodologies do not explicitly support making architectural changes after services are deployed and in execution. Among the SOSE methodologies that do provide some support for changing or even determining the service architecture at runtime, SeCSE allows the highest flexibility since it supports creating on-the-fly service compositions addressing specific runtime user requirements by directly interacting with runtime discovery process. Chang also supports adaptable services through dynamic service discovery and composition but detailed guidance is missing; moreover, Chang provides guidance of how to handle mismatches between requirements and discovered services.

SDLC, SOAM, SOAF, and TRUE do not directly suggest any techniques to handle changes but they, to certain extent, support making changes by emphasizing the importance of designing for adaptability.

Selection guideline:
Some enterprises adopt SOA because its potential ability of building dynamically adaptable SBAs. For these enterprises, whether a SOSE methodology provides strong support of handling changing requirements after services are published is of great importance. From our analysis the levels of such support range from design for adaptability to the determination of service architecture at runtime. Enterprises may use this criterion to select the one that meets their adaptability needs.

3.3.7 SC7 Runtime activities

Most of the SOSE methodologies focus on providing guidance on design time activities, only five methodologies explicitly describe activities that need to be performed at runtime. When looking at the objectivities (*GC1 Objective*) of the methodologies, we noticed that the ones that aim at design and analysis (such as OASIS and SOAD) naturally contain only design time activities, whereas the ones intend to provide guidance for the entire engineering process often take runtime activities into account. For instance, SeCSE, SDLC and SOMA covers both design time and runtime activities; different from SDLC and SOMA that only briefly mention some runtime activities (e.g. dynamic binding and monitoring in SLDC and monitoring and management in SOMA), SeCSE describes its runtime activities (e.g. runtime service discovery, composition, re-binding, monitoring) in detail.

The other two SOSE methodologies that describe runtime activities are Chang and SODIUM, both aiming at (dynamic) service composition. In Chang, a dynamic composition handler(DCH) is briefly introduced, which enables runtime service invocation, service adaptation and biding. However, detailed information about how a DCH actually works and how to use it is missing. In SODIUM, runtime service discovery and composition are described in detail; in addition, tools that support these runtime activities are also introduced.

Selection guideline:
Similar to criterion *SC6 Architectural change*, this criterion can be used to evaluate the runtime support provided by SOSE methodologies with the purpose of achieving dynamically adaptable SBAs.

3.3.8 SC8 Non-functional requirements

NFRs, including quality requirements (e.g. performance, security) and business requirements (e.g. business goals and vision), are of great importance to services. Most of the SOSE methodologies we evaluated (except for CBDI-SAE and OASIS) consider NFRs in their process, providing guidelines on collecting, specifying or achieving NFRs, with different levels of detail. In SOAD, some general principles or quality factors are identified and act as its baseline for design; however, how to collect, specify, and satisfy NFRs from the users of services is not discussed. In Chang, SOAF, SOUP, SOMA, SOSE, and TRUE, collecting and specifying of NFRs are mentioned as part of their engineering tasks; however, details about how to carry out these tasks are missing.

In SeCSE, SDLC, and SODIUM, NFRs are explicitly considered and discussed in detail. In SeCSE, binding rules and quality of service (QoS) constraints and objectives (that will be used at runtime by the binding and re-binding process)

are first defined at design time. After that, in requirement-based service discovery process mechanisms are provided for matching functional and quality of service requirements with published service requirements.

SDLC discusses how to capture NFRs as well as how to monitor and enforce them by means of service-level agreements (SLAs). SLAs are used to formalize the usage conditions and quality-level guarantees during the process of service negotiation process; and finally whether the behavior of services abides by the agreed SLAs are determined by the process of service monitoring.

In SODIUM, QoS are first specified in two parts: one part contains the absolute QoS constraints that are used to exclude services, and another part contains the optimization criteria that are used to rank the services; and then QoS are used to sort and select a ranked list of candidate services for further composition.

Selection guideline:
Some enterprises have higher NFRs than the others. For these enterprises, it is even more important to capture and specify NFRs collected in a systematic way and carry out specific activities to ensure the NFRs delivered by services or SBAs match the needs of their users.

3.3.9 SC9 Variability

To increase the reusability of services, a service should be provided with different quality attributes to fulfill the needs of different groups of service consumers. Unfortunately, our evaluation show that only Chang, SDLC, and SeCSE provide guidelines of designing and/or delivering services for specific service consumers according to their needs. In Chang, commonality and variability among different service consumers and context are analyzed and modeled. Non-functional mismatch is identified as a variation point in service design, and this information is used in mapping the NFRs collected from various service consumers and stakeholders to the candidate services.

Both SDLC and SeCSE describe the use of SLAs in specifying different requirements of service consumers and in reaching agreements with service providers on service consumption conditions. Further, SDLC emphasizes the importance of the design for service reuse and suggests making services more generic, abstracting away from differences in requirements between service consumers and attempting to provide the generic service in different context where it is applicable. SODIUM supports the variability of NFRs by allowing the users to specify individually the required QoS that must be offered by services. This information is used when selecting and prioritizing candidate services for service composition.

Selection guideline:
For enterprises that act as a service provider and aim at providing their services to different groups of users with different requirements, this criterion is crucial for selection.

3.3.10 SC10 Perspective

The engineering of SBAs is concerned with both service provision and service consumption processes. However, most of the SOSE methodologies, including OASIS, SDLC, SOAD, SOAF, SOMA, SOSE, SOUP, and TRUE, are designed from the perspective of service provision. These methodologies focus on collecting requirements from their clients, designing services and service-oriented architecture to fulfill these requirements, and publishing and delivering the services to service consumers. Most of them do not provide any guidelines of how to consume the published services, except for SDLC that briefly explains the discovery, composition, invocation, and monitoring of the published services.

Two SOSE methodologies, Chang and SODIUM, are designed from the perspective of service consumption. They focus on collecting requirements from end users, discovering and selecting services to fulfill these requirements, and composing the selected services into service compositions. Despite that they both focus on service consumption, SODIUM also mentioned briefly how to publish these service compositions whereas Chang does not provide any guidelines in this regard.

CBDI-SAE and SeCSE, are designed from the perspective of both service provision and consumption. CBDI-SAE consists of four disciplines, among which the provide and consume disciplines are considered equally important. Nevertheless, the consume discipline focuses on addressing business requirements and much effort has been put in collecting business requirements, improving business using services, and solution assembly. Guidance specifically on consumption-related activities such as service discovery and service negotiation is missing. In SeCSE, how to perform both the consumption and provision process has been explained in detail.

Selection guideline:
Knowing the perspective that a SOSE methodology is designed from is of great importance for enterprises to select the one that matches their business at best. Enterprises that provide services to market users may interested in the ones that focus on service provision; enterprise that integrate existing services to service compositions may be interested in the ones focus on service consumption; and enterprises that aim at building SBAs consists of both internally developed and

externally provided services may interested in the ones focus on both service provision and consumption.

3.3.11 SC11 multiple organizations

The SOSE process requires interaction and collaboration between multiple development roles, which might be distributed in multiple organizations. Unfortunately, most of the SOSE methodologies we evaluated do not provide sufficient guidance of how to handle the interaction, especially the collaboration between multiple organizations. CBDI-SAE, OASIS, and SDLC have mentioned the need for collaboration between multiple organizations, but details are missing. In CBDI-SAE, an organizational view of the SOSE process is suggested, illustrating the relationship between some example organization units (which could be located in multiple organizations as well). In this view, the tasks associated with each organization units (or organizations) are made explicit and their interaction with the tasks of other organization units (or organizations) are also given. For instance, an application delivery unit is responsible for solution assembly, which requires input from business architecture unit that is responsible for business requirements planning and improvement. OASIS emphasizes that creating service architectures is "all about creating a common dialog between the various different groups and deciding upon a boundaries that work across the business" and suggests a collaborative working with e.g. an intensive session or a conference. SDLC indicates that business domain is a functional domain comprising a set of current and future business processes that can collaborate with each other to accomplish a higher-level business objective, implying that collaboration is required between business domains.

SeCSE and SOAF provide relatively better guidance in this respect. In SeCSE, each engineering activity is associated with at least one development role. Depending on the description of the development role, it is easy to understand which roles might be in the same organization or different ones. For instance, a service developer and service architect might be both in an organization responsible for service provision, whereas an SBA architect might be in another organization responsible for service consumption. As the interactions between the engineering activities are described in detail, naturally the potential collaboration between the organizations becomes clear as well.

In SOAF, high-level collaboration models as well as the interactions, the sequence of activities and the work products exchanged between internal and external participants suggested to be well documented in the modeling phase; in the mapping and assessing phase, external entities need to be captured as part of the portfolio data. By explicitly distinguishing internal and external participants, work products, and business entities, the interactions needed between multiple organizations become explicit too.

Selection guideline:
Some enterprises develop services and SBAs for internal uses. Some other enterprises have wider business scope and often share services with their business partners (consuming services provided by the others or provide service to the others). Consequently, multiple parties have to collaborate to accomplish the engineering of SBAs. For these enterprises, this criterion is useful for selection.

4 Observations

The evaluation and analysis of the methodologies highlighted some weaknesses calling for further improvement. In the following, we shall discuss our observations on where the SOSE methodologies could be improved.

– Many service life cycle activities are not well supported. By evaluating the life cycle coverage (*GC2 Lifecycle*) of the SOSE methodologies, we noticed that some service life cycle activities have been supported by most of the SOSE methodologies, whereas some activities have received much less support. The lack of support for these activities shows that the existing SOSE methodologies focus mainly on the design and analysis part of the SOSE process, but pay little or not sufficient attention to the constructing, delivering and management part.
When constructing an SBA, ideally services can be dynamically discovered and composed at runtime, which often requires a well-functional service registry. As shown in Table 4 registry-related activities (selection, update, maintenance) are hardly supported by any of the SOSE methodologies. Without such support, an enterprise that intends to publish their services in a dedicated registry to increase service reusability and discoverability could face the risk of building up a service repository that is often outdated and does not function as expected.
When delivering an SBA, both service providers and their consumers would benefit from a service negotiation that ensures that both of them reach an agreement on the behavior of the services and on the price paid for using the services. As shown in Table 4, service negotiation is supported only by SeCSE. Without sufficient guidelines on service negotiation, both service providers and consumers face the difficulties when the services do not behave or paid as expected.
When managing an SBA, it is important that the behavior of the services can be monitored and the services can be adapted when required. Unfortunately, as shown in Table 4, many SOSE methodologies do not provide any

support in service monitoring and service management. Without keeping track of the behavior of published or consumed services, the manager of a SBA would not be aware of runtime changes in terms of e.g. quality of services. As a result, it would be difficult to identify the needs for substituting services when failure occurs. The quality of a SBA as a whole therefore cannot be ensured. In summary, the existing SOSE methodologies provide very weak support to aid the dynamicity of SBAs. In practice, services are often developed and shared only internally to an enterprise, which heavily hurdles their reusability to a wide range of consumers; services are often discovered manually at design time, which significantly decreases the efficiency of engineering SBAs; and services are often not able to dynamically adapt to new requirements or environmental changes, which obviously reduces the benefits that SOA promises.

– Development roles are significantly under-addressed. Surprisingly, many SOSE methodologies we evaluated do not provide any guidance on the development roles required in a SOSE process. As shown in Table 5, only three methodologies mentioned their development roles, among which only two methodologies associate the roles to the activities. Generally speaking, a methodology is a description of a process, explaining what and how (and who) to carry out a set of procedures. Especially when applying a methodology, *who* becomes one of the top questions from its users.
  In TSE, the development roles like software architect, designer, developer and tester are well known and their responsibilities are often clear to practitioners. In SOSE, the process is more complex than that of TSE, involving engineering, composition, continuous adaptation and consumption of services. Naturally, more development roles are required in that traditional development roles often split into two types of roles, one focusing on services and another focusing on SBAs (e.g. service designer and SBA designer, respectively). Moreover, some development roles are specific to SOSE and are not found in TSE; for example, service modelers, service monitors or adaptation designers are only found in SOSE. What makes it more complicated is that due to the separation of service consumption and provision and the shift in ownership of services, a development role may act from multiple perspectives simultaneously, each with different goals and competencies. For instance, a service designer with the perspective of a *service provider* is concerned with the identification of services to provide to others whereas the service designer with the perspective of a *service consumer* she is concerned with the identification of services for integration purposes.
  The complexity introduced in SOSE development roles results many different roles proposed in the literature. A common understanding of what types of SOSE roles are required and what their responsibilities are is currently missing. As a result, to increase the understandability and usability of a SOSE methodology, it is more important that it explains in detail which roles (with what skill) should be involved and associates the roles with the SOSE activities within the proposed SOSE process.
– The support of service consumption is significantly missing.
  The engineering of SBAs means both service provision and service consumption. From our evaluation of the existing SOSE methodologies, we noticed that most of them focus on the provision of services, aiming at designing, developing, and delivering services that meet the requirements of their users. Service consumption is often paid little attention to. However, in order to discover, select, and invoke these services, service consumers need guidance of how to carry out these tasks. As shown in Table 4 the activities carried out by a service consumer is only partially covered by less than half of the SOSE methodologies we evaluated. Similarity, as shown in Table 5, there are only two SOSE methodologies describe their process from the perspective of service consumption and two from both consumption and provision perspective, the other eight methodologies all consider the SOSE process as a service provision process.
  Not only service consumers need guidance on service consumption-related procedures, but also service providers need guidance of how to design and develop services that can be consumed by many different groups of service consumers with different requirements. However, many SOSE methodologies designed from the perspective of a service provide only cover the service provision process but completely leave out the aspects related to service consumption. For instance, as shown in Table 5, only SDLC and SeCSE designed from the perspective of service provision address the variability of non-functional requirements from different groups of service consumers. Regardless of the perspective that a SOSE methodology takes, service consumption is part of the SOSE process and should be considered as equally important as service provision. A SOSE methodology that specifically addresses service provision should not consider the provision process in isolation, but take into account service consumption aspects, such as discoverability, composability, negotiability, and reusability.

5 Related work

The need for comparing and evaluating existing SOSE methodologies has been recognized and initial research has been performed. In the survey of SOSE methodologies conducted

by Ramollari et al. [11], a set of characteristics of SOSE methodologies has been used as their evaluation criteria. All of these criteria have been directly or indirectly covered in our evaluation framework. Criteria such as *lifecycle coverage, existing process&techniques*, and *applied in industry* can be directly mapped to our criteria: *GC2 Lifecycle*, *GC11 Integration*, and *GC9 Maturity*. Some other criteria such as *delivery strategy* and *UML* Are addressed by *SC3 The creation of services* and *GC3.b Notations* in our framework. There is one difference in interpreting between criteria *consumer view & provide view* in Ramollari et al.'s survey and our criterion *SC10 Perspective*: in the survey, a consumer view refers to declarative and business process oriented development through service composition and a provider view refers to programmatic and component oriented development; in our work we differentiate the consumption and provision process by 'direction of operations' on services, either in (i.e. use) or out (i.e. delivery). The main reason is that a service provider may carry out business process oriented development and provide composite services to consumers, and a service consumer may carry out programmatic development process to integrate consumed services to service compositions or SBAs. Our interpretation of service consumption and provision is of wider scope than that of Ramollari et al.

An overview of SOSE methodologies is given by Kontogogos and Avgeriou [12]. In their overview, six criteria have been used for analysis. All of them have been addressed in our evaluation framework. Criteria *lifecycle, adaptability,* and *industry* are directly covered by *GC2 Lifecycle*, *SC6 architectural change*, and *maturity*. Criterion *detail* is addressed by multiple criteria (such as *GC3.a Artifacts, GC4 Procedure, GC5 Principles, GC10 Management*) in our framework to assess the degree of detail including outputs, tasks, guidelines, deliverables provided by SOSE methodologies. Although criteria *service description* and *behavior specification* are addressed by our criteria *GC3.a Artifacts* and *GC3.b Notations*, they can be used to assess in detail the specification of services (as part of the artifacts). Due to the important role played by services, we consider these two criteria as being complementary to our framework.

Both works discussed above do not analyze and evaluate the service-specific features that SOSE methodologies should entail. Similar to our approach, a criteria-based evaluation framework for SOSE methodologies proposed by Gholami et al [24] consists of generic and service-specific criteria. The generic criteria are in general similar to the ones that we proposed. The major difference is that we assess the coverage of life cycle phases more precisely by looking into the activities of each life cycle phase, whereas Gholami et al. only provide an overall score showing the degree of methodology support.

The service-specific criteria proposed by Gholami et al.'s framework and ours are significantly different. In Gholami et al.'s framework the criteria are derived from two sources: one is service-related activities (e.g. business, modeling, service-oriented analysis & design, service testing, SLA monitoring), another is quality requirements (e.g. service agility, adaptable with legacy systems, process agility). Assessing the support of service-related activities and quality requirements are definitively relevant. Nevertheless, Gholami et al.'s framework does not include all the service-related activities. For instance, service negotiation, requirements engineering from external and internal users and service registry-related activities are missing. Restricting service-specific criteria only on these two aspects limits the identification of other important features that SOSE methodologies should support, such as dealing with uncertainties, explaining development roles and their responsibilities, collaboration between multiple organizations, and the awareness of provision and consumption process. In our work, instead, we derived the service-specific criteria from a list of differences between SOSE and TSE identified from a systematic literature review. As such, we are confident that our service-specific criteria are a set of must-have service-oriented features for SOSE methodologies.

6 Conclusions

An enterprise that decides to adopt SOA often needs to choose a SOSE methodology to follow or to tailor for its own usage. Given many SOSE methodologies available in the literature, the question is how to select one that is both beneficial and feasible to the enterprise. The evaluation framework proposed in this work helps the enterprise in deciding how to choose a SOSE methodology and tailor it for its own usage.

For understanding the characteristics of SOSE methodologies and gaining insight into their support for service orientation, in this work we took a feature analysis approach and derived a set of generic- and service-specific criteria for comparing the existing SOA methodologies. Enterprises usually select a SOSE methodology to fulfill some specific needs. To support selection, these needs can be associated with the criteria of our framework.

The framework has been validated by analyzing 12 SOSE methodologies. Next to their comparison, this exercise helps differentiating the methodologies that are truly service-oriented (e.g. SeCSE and SDLC) from those that deal little with service aspects (e.g. SOAD and SOUP). As an additional result, we extracted some guidelines that can further support selection.

The analysis of the SOSE methodologies also highlighted some weaknesses calling for further improvement: increasing the coverage of service life cycle activities, improving the support for development roles, and emphasizing the guidance on service consumption-related issues.

Lastly, no one SOSE methodology covered two criteria, *GC6 Formality* and *GC7 Specialization*. This can be due to various reasons that could be argued upon. This lack of coverage may identify either a gap in the state of the art or that our choice of criteria is too demanding, calling for further investigation in the future.

Whereas no one SOSE methodology we analyzed fulfills the complete set of criteria, we cannot objectively assign higher of lower priorities to any of them. In principle, an "ideal" SOSE methodology would cover all features we enlist in our evaluation framework. However, in practice enterprises that need to select SOSE methodologies can use (a subset of) these criteria (depending on their needs) to evaluate SOSE methodologies available for use.

Acknowledgments The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

Appendix A: The judgment scale to assess the methodology support for a feature

See Table 6.

Table 6 The judgment scale to assess the methodology support for a feature

Generic scale point	Definition scale point	Scale point mapping
No support	Fails to recognize. The feature is not addressed in the methodology.	–
Implicitly support	The feature is addressed implicitly. It can be recognized in terms of interpretation.	+
Explicitly support	The feature is explicitly addressed in the methodology, but no detailed information is given.	++
Strong support	The feature is explicitly addressed in the methodology with detailed information or guidance.	+++

Appendix B: The service lifecycle model

See Table 7.

Table 7 The phases of the service lifecycle model and their description (Based on [23])

Role	Life cycle phases	Description
Service provider	Market scan	Investigate the market demands and orient the service production.
	Registry update	Update the registry when new services are published or existing services are changed.
	Registry maintenance	Maintain the consistency, efficiency of the service registry to assist service discovery.
Service consumer	Service discovery	Look up published service candidates that comply with the requirements
	Service composition	Ensure that services in an application can be well assembled to work together to fulfill business requirements.
	Service negotiation	Exchange a number of contract messages with a service provider in order to reach an agreement to access the services.
	Service invocation	Each of the services that are discovered and composed in the earlier activities has to be invoked for execution.
	Service monitoring	Keep track of the behavior of the invoked services.
Application builder	Requirements engineering	Analyze the objective and functionality of SBAs, collect requirements and needs from the end users.
	Application design, application implementation and module testing	Similar to the ones in traditional software engineering.
	Application testing	Ensure that the integrated services perform as expected.
	Application maintenance	Take care of the changes that might occur from the end user requirements or the behavior of the composed services.

References

1. Zimmermann O, Krogdahl P, Gee C (2004) Elements of service-oriented analysis and design
2. Arsanjani A, Ghosh S, Allam A, Abdollah T, Gariapathy S, Holley K (2008) SOMA: a method for developing service-oriented solutions. IBM Syst J 47(3):377–396
3. Mittal K (2006) Build your SOA, part 3: the service-oriented unified process. Technical report, IBM developerWorks
4. Jones S, Morris M (2006) A methodology for service architectures. http://www.oasis-open.org/committees/download.php/15071
5. Allen P (2007) SOA best practice report: the service oriented process. Technical report, CBDi Journal
6. Papazoglou MP, Heuvel WJvd (2006) Service-oriented design and development methodology. Int J Web Eng Technol 2(4):412–442
7. Chang SH (2007) A systematic analysis and design approach to develop adaptable services in service oriented computing. IEEE congress on services, pp 375–378
8. Kitchenham BA (1996) Evaluating software engineering methods and tool part 1: The evaluation context and evaluation methods. SIGSOFT Softw Eng Notes 21(1):11–14
9. Jayaratna N (1994) Understanding and evaluating Methodologies: NIMSAD, a systematic framework. McGraw-Hill Inc, New York
10. Siau K, Rossi M (1998) Evaluation of information modeling methods—a review. In: Proceedings of the thirty-first annual Hawaii international conference on system sciences, vol 314. IEEE Computer Society, Washington
11. Ramollari E, Dranidis D, Simons A (2007) A survey of service oriented development methodologies. In: Proceeding of second European young researchers workshop on service oriented computing. University of Leicester, Leicester, pp 75–80
12. Kontogogos A, Avgeriou P (2009) An overview of software engineering approaches to service oriented architectures in various fields. In: Proceedings of the 18th IEEE international workshops on enabling technologies: infrastructures for collaborative enterprises, IEEE Computer Society, pp 254–259
13. Karam GM, Casselman RS (1993) A cataloging framework for software development methods. Computer 26(2):34–45
14. Gu Q, Lago P (2009) On service-oriented architectural concerns and viewpoints. In: Proceeding of 8th working IEEE/IFIP conference on software architecture. IEEE, Cambridge, pp 289–292
15. Maximilien EM, Singh MP (2004) Toward autonomic web services trust and selection. In: Proceedings of the 2nd international conference on service oriented computing. ICSOC '04, ACM, pp 212–221
16. Kajko-Mattsson M, Lewis G, Smith D (2007) A framework for roles for development, evolution and maintenance of SOA-Based Systems. In: Proceedings of the international workshop on systems development in SOA environments, 7
17. Zimmermann O, Mueller F (2004) Web services project roles. IBM DeveloperWorks Article
18. Karhunen H, Jantti M, Eerola A (2005) Service-oriented software engineering (sose) framework. In: Proceedings of international conference on services systems and services management, vol 2, pp 1199–1204
19. SeCSE (2007) SeCSE methodology version 3. Technical Report A5.D4.2, The SeCSE Consortium. http://www.secse-project.eu
20. Erradi A, Anand S, Kulkarni N (2006) SOAF: An architectural framework for service definition and realization. IEEE Int Conf Serv Comput, pp 151–158. doi:10.1109/SCC.2006.97
21. Topouzidou S (2007) SODIUM, service-oriented development in a unified framework. http://www.atc.gr/sodium/upload/publications/SODIUM_FinalReport.pdf (accessed 16.11.10)
22. Engels G, Hess A, Humm B, Juwig O, Lohmann M, Richter JP, Voß M, Willkomm J (2008) A method for engineering a true service-oriented architecture. In: ICEIS (3–2), pp 272–281
23. Gu Q, Lago P (2007) A stakeholder-driven service life cycle model for SOA. In: Proceeding of 2nd international workshop on Service oriented software engineering. ACM, Dubrovnik, pp 1–7
24. Gholami MF, Habibi J, Shams F, Khoshnevis S (2010) Criteria-based evaluation framework for service-oriented methodologies. In: Proceedings of the 12th international conference on computer modelling and simulation. UKSIM '10. IEEE Computer Society, Washington, pp 122–130