Softw Syst Model (2014) 13:361–390
DOI 10.1007/s10270-011-0222-z

REGULAR PAPER

Enhancing the OPEN Process Framework with service-oriented method fragments

Mahdi Fahmideh Gholami · Mohsen Sharifi · Pooyan Jamshidi

Received: 23 April 2011 / Revised: 5 October 2011 / Accepted: 28 October 2011 / Published online: 16 November 2011

Abstract Service orientation is a promising paradigm that enables the engineering of large-scale distributed software systems using rigorous software development processes. The existing problem is that every service-oriented software development project often requires a customized development process that provides specific service-oriented software engineering tasks in support of requirements unique to that project. To resolve this problem and allow situational method engineering, we have defined a set of method fragments in support of the engineering of the project-specific service-oriented software development processes. We have derived the proposed method fragments from the recurring features of 11 prominent service-oriented software development methodologies using a systematic mining approach. We have added these new fragments to the repository of OPEN Process Framework to make them available to software engineers as reusable fragments using this well-known method repository.

Keywords Service-oriented software development · OPEN Process Framework · OPF repository · Method fragment · Situational method engineering

Communicated by Prof. Brian Henderson-Sellers.

Preliminary contributions of authors on the proposed subject matter of this paper presented in conferences are as follows: (1) Fifth IEEE International Conference on Research Challenges in Information Science (RCIS'11), France, 19–21 May 2011. (2) Second International Conference on Software Engineering and Computer Systems (ICSECS'11), Malaysia, 27–29 June 2011. (3) 12th International Conference on Computer Modeling and Simulation (UKSim-AMSS), UK, 24–26 March 2010. (4) European Modeling Symposium (EMS'10), Pisa, Italy, 17–19 November 2010.

M. F. Gholami · M. Sharifi (✉)
School of Computer Engineering,
Iran University of Science and Technology,
Tehran, Iran
e-mail: msharifi@iust.ac.ir

M. F. Gholami
e-mail: m.fahmideh@comp.iust.ac.ir

P. Jamshidi
School of Computing, Lero-The Irish Software Engineering Research Centre, Dublin City University, Dublin, Ireland
e-mail: pooyan.jamshidi@computing.dcu.ie

1 Introduction

Software engineers are currently faced with increasing demands for the development of software systems that are heterogeneous, geographically distributed and dynamic in nature in the sense that system components can be dynamically detached, added, or reconfigured at runtime [1]. Service-oriented paradigm has provided the basic concepts and means for development of such software systems. Services as fundamental elements of service-oriented systems play a pivotal role in service-oriented software development. They are self-contained, loosely coupled, platform independent, stand-alone, and autonomous elements that form the underpinning of service-oriented systems [2]. A number of available published services can be composed together to form a large software system. Services collaborate via standard message protocols in a loosely coupled distributed heterogeneous environment. It is thus possible for software engineers to develop service-oriented software systems via composition of discovered services during software construction or execution rather than crudely following traditional phases of analysis, design, and implementation. To take advantage of existing services, service-oriented software developers must perform extra tasks compared to traditional software developers that are specific to service orientation. Furthermore,

software requirements are less known to service-oriented software developers, while traditional software developers have more knowledge about software requirements at earlier stages of software development and know the tasks they must perform to satisfy these requirements earlier [3].

Service-oriented *software development methodologies* (*SDM*s) have tried to identify tasks that service-oriented software developers must carry out in addition to tasks carried out in traditional software development methodologies. These extra tasks are specific to *service-oriented software development* (*SOSD*). Although SDMs have some common features (e.g., cover the same life cycle phases), they have been proposed for different purposes, ranging from project management to system modernization, and from business analysis to development of technical solutions [4]. Given the variety of existing SDMs, it is hard for software engineers to decide which SDM fits best the specific needs of a project. Furthermore, specific SOSD tasks in service-oriented SDMs are tightly interwoven with traditional tasks making it very hard for developers to extract and assemble the required SOSD tasks in support of requirements of a specific project. This asserts the evidence that there is no universal software development process[1] that is appropriate for all situations [5–8]. Some of the issues that developers must consider for every situation include organizational maturity and culture, people skills, commercial and development strategies, business constraints, and tools [9,10]. They must therefore construct their own project-specific SDM or software process for the development of their software.

One of the well-known approaches for tailoring SDMs is *situational method engineering (SME)*, wherein a project-specific SDM is constructed from *reusable method fragments* [11,12] or *method chunks* [7,13]. To allow the construction of a wide range of project-specific SDMs by developers and method engineers, a repository of method chunks is necessary [5]. An established approach in line with the ideas of SME is the *Object-oriented Process, Environment, and Notation* (*OPEN*) [14,15]. OPEN has a repository of reusable method fragments called *OPF*, from which method engineers can select method fragments using suitable construction guidelines. They can then assemble their selected fragments to construct a wide spectrum of project-specific SDMs based on the unique set of requirements of SDMs. Existing method fragments in OPF can be used in the construction of many types of situational SDMs except for service-oriented SDMs. In other words, one of the main shortcomings of OPEN is its lack of support for SOSD. Existing method fragments in OPF repository are mainly intended for *object-oriented* (*OO*) software development, while method fragments in support of agility and aspect orientations are also forthcoming [36–39]. Although there are many commonalities between OO software development and SOSD, they have many differences too requiring new method fragments in support of SOSD.

Motivated to enhance OPF repository, we propose a new set of method fragments in this paper in support of SOSD in conformance with the underpinning metamodel standard of OPEN [27] using our previous systematic approach [16]. We have designed these method fragments in such a way as to facilitate the engineering of service-oriented SDMs based on OPEN. To do so, we studied the SOSD literature, specifically the development processes of most well-known existing service-oriented SDMs, extracted their recurrent tasks, and presented the extracted tasks in the form of method fragments. OPEN CASE tools [17] that manage the OPF repository can import the proposed method fragments as extensions to their existing OPF repository and use them to construct project-specific service-oriented SDM.

The main audiences of our research reported in this paper are those specific groups of software developers who are *method engineers* or *process engineers*. Generally, method engineers are responsible for constructing, tailoring, and maintaining software processes for use in a wide range of software projects in a software development organization. In the realm of service-oriented systems, method engineers need a set of domain-specific method fragments, as reusable building blocks of methodologies, to assemble method fragments together and construct a new project-specific service-oriented methodology. Notwithstanding the multitude of service-oriented development methodologies, the lack of knowledge about service-oriented software development in a well-structured and standard format has long been felt. The proposed method fragments, as methodological knowledge, provide support for method engineers to create knowledge on developing service-oriented systems and share it with other method engineers. Fortunately, OPEN is a good candidate because it provides a standard meta-model for representation of methodological knowledge via autonomous and coherent method fragments.

In addition, from a method engineer's point of view, the authors suppose that contributed method fragments represent pivotal activities, rather than traditional software engineering activities and practices. The proposed fragments must be incorporated into the software development process when an inherently complex and dynamic distributed system is being developed and maintained in a service-oriented style. It is generally agreed today that method fragments can capture and represent the knowledge on software processes in a well-structured and reusable format.

Having delineated the outline of our research, we have organized the rest of the paper as follows. Section 2 presents the basic concepts underlying our research. Sec-

[1] We have used the terms *method, methodology, software development methodology, and software development process* synonymously in this paper.

tion 3 presents a brief review of prominent service-oriented SDMs that have been selected as main sources to define new method fragments. Section 5 explains the way in which appropriate method fragments have been constructed. Section 6 presents our proposed method fragments. Section 7 identifies the position of these method fragments in the OPEN process framework. Section 8 presents a discussion on the proposed method fragments. Section 8 shows the applicability of the new method fragments through presentation of a partial case study. Section 9 concludes the paper and presents further extensions to the reported research.

2 Background

In this section, we briefly review the main concepts underlying our proposition in this paper.

2.1 Situational method engineering

The prevalent belief that no single software development process can be applicable to all situations is the main reason for the emergence of *method engineering* (*ME*). ME was first introduced by Kumar [5] as a software engineering discipline aimed at constructing a project-specific software development process to meet given organizational characteristics and project situations. Brinkkemper [6] elaborated ME definition later to: "The engineering discipline to design, construct, and adapt methods, techniques and tools for the development of information systems". The most well-known subset of ME, namely SME, is concerned with the construction, adaptation, or enhancement of a suitable SDM for the project at hand instead of looking for a universal or widely applicable one [5–8]. In the SME approach, an SDM is constructed from a number of encapsulated and fragmentized methods stored in a repository. Typically, a method engineer goes through the following SME steps to construct a project-specific SDM [19]:

1. Elicitation and specification of specific requirements of target SDM.
2. Selection of a number of most relevant method fragments from the repository based on a number of situational factors highly specific to the particular software development organization and particular situation of the project.
3. Assembly of the chosen method fragments to form a coherent project-specific SDM.

Method engineers can use *computer aided method engineering* (*CAME*) tools to do the above four steps for saving, restoring, selecting, and assembling method fragments [21]. One instance of the SME approach that is highly compatible with the above steps and is extensively used in the development of a wide range of software project types, especially in the OO context, is the OPEN Process Framework [14,15]. Industrial use of OPEN demonstrates its viability in software development [22] so much so that we have been motivated to base our research on OPEN. In the next section, we present OPEN in more depth.

2.2 OPEN Process Framework as a foundation for SME

OPEN is the oldest established software development process introduced in 1996 as a result of the integration of three second-generation OO software development SDMs, namely MOSES [232023], SOMA [24], and Firesmith. OPEN is known as one of the most popular software development processes with support for full life cycle. OPEN has been updated recently to be in conformance with ISO/IEC 24744 [25], which is mainly intended for use in the development of software systems or in the construction of project-specific SDMs based on projects' circumstances. A not-for-profit consortium comprising an international group of methodologists, academics, and CASE tool vendors maintains OPEN [26]. OPEN contains an underpinning metamodel (a model for describing method fragments or software development processes), a rich repository of method fragments, and several kinds of usage guidelines that explain how method engineers can use method fragments. To construct a project-specific SDM, a method engineer selects his/her required method fragments from the OPF repository, wherein each method fragment is an instance of OPEN metamodel (Fig. 1). Given our objective in this paper, we study the metamodel of OPEN and its OPF repository in more detail here.

2.2.1 Metamodel

The metamodel of OPEN provides a clear way for formally representing method fragments such as phases, processes, tasks, techniques, work products, and roles. It is imperative that each method fragment should conform to the OPEN metamodel standard. This implies that new method fragments extending the repository must conform with the metamodel too. It should be noted that the underpinning OPEN's metamodel has been updated and aligned with the ISO/IEC 24744 metamodel. This standard metamodel incorporates experience from earlier SME and is used to represent SDMs [25]. In this paper, we have used the recently updated OPEN metamodel terminology. Having the recent update of OPEN metamodel with ISO/IEC 24744 in mind, the five core classes of method fragments are as follows (Fig. 2) [14,15]:

1. *WorkUnitKind* Operations that should be performed by persons or tools to develop the required WorkProductKind. WorkUnitKinds are categorized in three levels of abstraction:

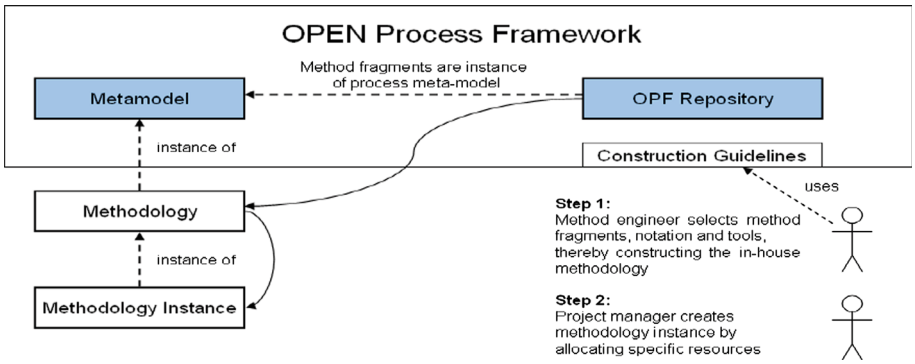

Fig. 1 Construction of a project-specific SDM from OPEN's metamodel (adopted from [27])

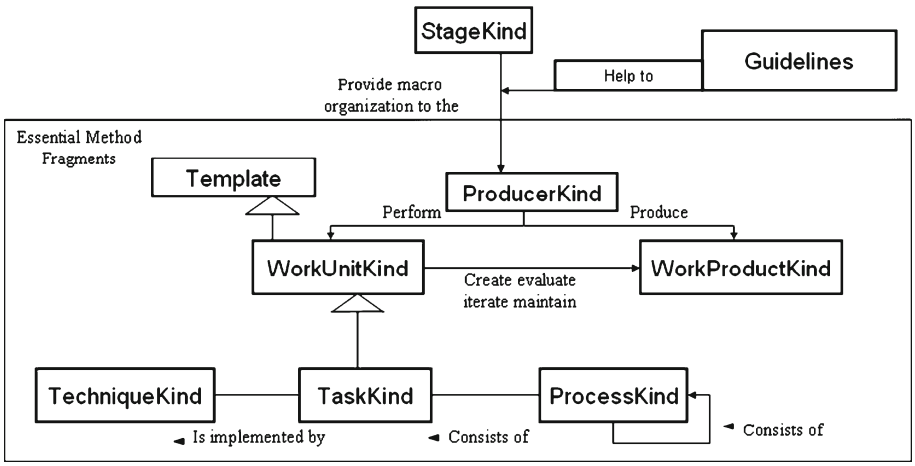

Fig. 2 Constituents of OPEN's Metamodel based on ISO/IEC 24744 terminology (adopted from [9])

- *ProcessKind* ProcessKind (called *Activity* in the older version of OPEN) is a coarse-grain type of typical WorkUnitKind consisting of a cohesive collection of TaskKinds that produces a related set of WorkProductKinds. In other words, a ProcessKind includes a group of relevant TaskKinds. Sometimes, ProcessKind has been referred to as a software engineering discipline.
- *TaskKind* TaskKind is a fine-grain type of WorkUnitKind consisting of a cohesive collection of steps that produce WorkProductKind(s).
- *TechniqueKind* TechniqueKind is an explicit set of procedures that explain how a TaskKind should be performed

2. *WorkProductKind* WorkProductKind is any significant produced artifact such as a diagram, a graphical or textual description, or a program produced during software development.
3. *ProducerKind* Persons or tools that develop expected WorkProductKinds are ProducerKinds.
4. *Language* Language is used to represent the produced artifacts using a modeling language, such as Unified Modeling Language (UML) [28], Object Modeling Language (OML) [29], or an implementation language.
5. *StageKind* StageKind is intended for use in defining the overall macro-scale and time-box of a set of cohesive WorkUnitKinds during the enactment of an instantiated OPEN. The whole instantiated process is structured temporally by the use of StageKind concept element.

2.2.2 OPF repository

Besides the metamodel, OPEN contains a large number of method fragments having different levels of granularity (ProcessKinds, TaskKinds, and TechniqueKinds) stored in a repository. OPF recommends the use of the Deontic matrix approach [13] for selecting method fragments from a repository. A two-dimensional Deontic matrix represents possible relationships between each pair of method fragments in the OPF repository. According to the five classes of the OPF's method fragments, possible meaningful combinations are as follows [30]: ProcessKind/TaskKind, TaskKind/TechniqueKind, ProducerKind/TaskKind, TaskKind/Work ProductKind, ProducerKind/WorkProductKind, and WorkProductKind/Language. For each cell of the matrix, a five-scale value can be assigned—M: Mandatory, R: Recommended, O: Optional, D: Discouraged, and F: Forbidden. Processes can be considered as

Table 1 Deontic Matrix showing the possible relations between the Requirements Engineering Process and its relevant TaskKind method fragments (adopted from [35])

TaskKind	Requirements Engineering
Develop BOM	O
Identify context	R
Conduct market research	O
Create white site	O
Identify user requirements	M
Define problem and establish mission and objectives	O
Establish user DB requirements	O

traditional activities in software development process, such as the *Design software architecture* process, which includes a number of cohesive tasks such as *Evaluate software architecture, Select software architectural patterns, Develop initial software architecture, Document relevant software architecture views*, and *Realize quality attributes* [31]. To fill in the cells of the matrix with expected values, the method engineer should consider many situational factors such as project size, skills of the development team, organizational culture, and usage context of the target SDM. For instance, Table 1 shows a part of the decision-making process of assigning enumerated values in Deontic matrix in a small B2C (business-to-customer) system [35]. The method engineer assigns possible values to make a mapping between requirements engineering process and its fine-grain requirements engineering tasks. Having situational factors of the project in mind, the method engineer decides that *Identify user requirements* is mandatory for the method (denoted by M). In contrast, the Identify context task is considered as optional (denoted by O) while Conduct market research is recommended (denoted by R). The method engineer can decide similarly if other processes and task method fragments are mandatory or optional. OPEN metamodel and OPF repository of method fragments provide the means for SME. The OPF repository provides reusable method fragments as well as well-known and traditional processes and tasks for the construction of project-specific SDMs.

3 Related work

OPEN has aimed to support the construction of SDMs in the manifold spectrum of software development. Over the years, several researchers have provided extensions to OPF in support of different software development approaches. Henderson-Sellers et al. have done significant work in enhancing OPF. They have added supportive method fragments to facilitate situational software process construction for different approaches of software development as listed below:

- *Extension for component-based development (CBD) support* Henderson-Sellers [33] has enriched OPF by specific method fragments to support situational software process construction for component-based software development.
- *Extension for Web-based software development support* Concerned with the characteristics of Web development, Haire et al. [34,35] have added a number of reusable method fragments to the repository for Web-based software development.
- *Extension for aspect-oriented programming* (AOP)*Support* Given that AOP aims to modularize crosscutting concerns of software development into a cohesive structure, Henderson-Sellers et al. [36] have added new method fragments in support of AOP to the traditional development method fragments of OPF.
- *Agent-OPEN* [37] In this work, a number of new method fragments have been proposed to support agent-oriented software development. The OPF repository has been integrated with agent concepts. The assortment of specific agent-oriented method fragments can be found in [37].
- *Extension for security support* Henderson-Sellers et al. [38] have presented a set of security-focused method fragments that have been extracted from the agent-oriented secure TROPS [40,41] methodology and added to the OPEN repository.
- Other supports for organizational transition [42,43] and usage-centered design [44] have been added to OPF too.

Although OPF has matured, and contains method fragments in support of various approaches such as OO, CBD, AOP, and Agent-OPEN development, we have identified deficiencies in the current OPF support for SOSD [45]. A major problem in SOSD arises when method engineers decide to construct a project-specific service-oriented development process. While the tendency for the development of service-oriented software systems and consequently appropriate service-oriented SDMs have received much attention [46,47], we investigated the current OPF method fragments and found no support for defining specific method fragments for SOSD [45]. For instance, identifying services from business processes, utilizing existing functionalities of legacy systems, and discovering required services published on the Web are only a number of concerns that force to boost OPF in favor of SOSD.

Aiming at resolving the above shortcoming, we have enhanced the OPF repository with new method fragments in support of service-oriented development processes. To do this, we studied service-oriented development challenges [46] and current prominent service-oriented SDMs that

prescribe successive systematic processes and tasks to handle service-oriented issues [48]. We have then explored service-oriented SDMs and extracted a set of processes and tasks [16] as method fragments for SOSD in conformance with the standard format of the metamodel of OPEN, so that they can be easily imported into OPF tools.

4 Service-oriented SDMs: appropriate sources for derivation of new method fragments

Service orientation is currently appraised as a favorable approach in which services are utilized as fundamental elements to develop distributed software systems. Services are realized via Web-service [50] technologies. Web Services are independent, self-contained, reusable, and loosely coupled computational elements that form the underpinning of service-oriented systems. They collaborate via standard message protocols in a loosely coupled distributed heterogeneous environment. Therefore, a number of available published services can be composed together to develop a large software system.

In these regards, academia, industrial practitioners, and grey literature such as white papers or technical reports SOSD approach have emerged recently in addressing huge issues of service-oriented software systems such as *Service identification, Service specification, Service realization, Service discovery, Service composition* and *Dynamic reconfiguration*, and *Service governance* [46,47,51]. Service-oriented SDMs provide systematic processes, guidelines, and techniques required for handling of these issues. All of these end-to-end SDMs use existing traditional software engineering processes with some enhancements that are exclusive to SOSD. We briefly describe notable existing service-oriented SDMs here. The major criteria for our selection of these SDMs include their successful applications in real projects, their high maturity levels, their high rates of citations, better accessibility to their resources, and their better documentations. A comparative study of existing service-oriented SDMs can be found in [47–49].

- *IBM SOMA* [52] In its original form, SDM included three main phases for identification, specification, and realization of services. Arsanjani et al. expanded this to seven phases including business modeling and transformation, solution management, identification, realization, specification, deployment/monitoring/management, and implementation/build/assembly. SOMA is the most well-known SDM for SOSD due to its good features of software development process, such as having an iterative-incremental process model, an architecture-centric development, and fractal modeling. SDM has been applied to several industrial projects successfully, so that it has been designed originally from experiences of developing hundreds of real service-oriented software systems.
- *SUN SOA Repeatable Quality (RQ)* [53] This SDM has been proposed by SUN Microsystems Corporation based on Rational Unified Process (RUP) [54] and eXtreme Programming (XP) [55,56] principles that have proven mature development processes. RQ contains five phases, namely inception, elaboration, construction, transition, and conception. These phases can be performed in an iterative-incremental, architecture and use-case centric development model. The applicability of RQ suffers from the lack of supportive documents describing details of internal process of SDM.
- *CBDI-SAE Process* [57] This SDM is part of the CBDI-SAE SOA *Reference Framework* (*RF*) introduced by the CBDI forum. It has four key phases, namely manage, consume, provide, and enable, which fully cover service-oriented development process.
- *MSOAM* [58] MSOAM focuses only on service-oriented analysis and design phases. Its fully documented process prescribes systematic tasks and guidelines to develop appropriate services at different levels of granularity. However, it stops at the beginning of the implementation phase.
- *IBM RUP for SOA* [59] This iterative SDM has added service-oriented contents and specific process to RUP. In this variant of RUP, three phases of identification, specification, and realization have been added.
- *SDM proposed by Papazoglou* [51] Papazoglou et al. have presented a detailed service-oriented SDM that comprises eight distinct phases, namely planning, analysis and design, construction, testing, provisioning, deployment, execution, and monitoring. Each phase is based on a number of principles and guidelines required for SOSD.
- *IBM SOAD (Service-Oriented Analysis Development)* [60] SOAD's process has resulted from combining *Business Process Modeling* (*BPM*), *Object-Oriented Analysis and Design* (*OOAD*) and *Enterprise Architecture* (*EA*) practices, techniques, and a number of suggested guidelines for identifying and modeling the right services. SOAD's process is rather cursory and does not fully cover service-oriented life cycle so that it would be better called as a service-oriented analysis and design technique rather than a holistic SDM. Its applicability is limited, so that one can only use its specific guidelines during SOSD.
- *Service-Oriented Unified Process (SOUP)* [61] SOUP is a hybrid SDM engineered from RUP along with XP for the development of service-oriented systems. It has six main phases, namely incept, define, design, construct, deploy, and support, in which early stages of software development look similar to those of RUP. Consequently, it has a heavyweight process and full documentation. When system becomes operational at the user environment, XP

principles and practices are applied. These latter phases form a lighter process during maintenance.

- *SDM proposed by Chang and Kim* [62] The SDM contains five phases, namely, identifying business processes, defining unit services, discovering services, developing services, and composing services.
- *Steve Jones' Service Architectures* [63] This SDM is based on the idea of decomposing business processes of organizations into business services resulting in business service architectures of organizations. It has a top-down view on organizations to get a set of business level services without their complete definition and implementation.
- *Service-Oriented Architecture Framework (SOAF)* [64] This SDM comprises of a set of tasks, techniques, and guidelines that are grouped in five phases to address service identification and to help in deciding on service granularity while integrating existing legacy systems. Its phases are information elicitation, service identification, service definition, service realization, roadmap, and planning.

In the next section, we explain how our new method fragments have been extracted from the above-enumerated service-oriented SDMs.

5 Methods of identifying reusable method fragments

There are two main alternative approaches for constructing method fragments [65]: *existing method re-engineering* and *ad hoc construction*. The first approach focuses on identifying and using method fragments from existing SDMs in a plug-and-play format. However, the ad hoc construction approach uses real industrial projects to construct method fragments when there is no explicit defined SDM. In the latter approach, constructed method fragments are evaluated in practice; they are considered as reusable method fragments when quality standards are satisfied. Constructed method fragments in both approaches can be added to the repository of method fragments. Existing method re-engineering approach is a suitable approach to obtain method fragments when method fragments can be extracted directly from existing proven and matured SDMs. If an SDM has a successful profile in industrial realm, it is reasonable to select it as a candidate and use it for constructing method fragments. Therefore, given the existence of many SDMs in the domain of SOSD, we thought it is reasonable to use them as our main source to construct our new method fragments.

By having these enumerated SDMs in mind while constructing, new reusable method fragments are re-engineered from them. To do this, we needed an explicit technique to identify method fragments from SDMs. It should be noted that each service-oriented SDM supports different processes. Interestingly, most SDMs prescribe different tasks with different names and ambiguous and non-standard terminologies that are in fact similar. They have the same tasks in mind, but from different viewpoints. If we consider them collectively in an abstract view, we can find out recurring tasks in their development processes. It is thus important to note that the multiplicity and similarity of tasks in service-oriented SDMs should be managed in some way during construction of method fragments to derive non-redundant and pure service-oriented related method fragments. The concept of process pattern can be utilized for this purpose. Process patterns are classes of common successful practices and recurring tasks (or process chunks) in SDMs [66]. In service-oriented SDMs, constituent development processes prescribe the same tasks, but with different names too. Given the lack of any full-fledged technique for extraction of process patterns from service-oriented SDMs, we have previously developed a systematic technique for analyzing and mining existing service-oriented development SDMs in terms of meaningful process patterns (for more detail, see [67]). We have proposed several strategies to help method engineers identify process patterns. By focusing on contemporary service-oriented SDMs, we extracted a comprehensive set of process patterns that were refined and gradually completed. Process patterns were completed and fixed when the analysis of SDMs identified no new process pattern as a candidate method fragment. Finally, we represented extracted process patterns in an existing standard repository of method fragments, namely the OPEN metamodel. Figure 3 shows our steps of extracting method fragments from service-oriented SDMs.

SOSD only expands existing traditional software development tasks and it is mainly considered as an evolution rather than a revolution. It can however be viewed differently as an approach to develop software out of method fragments that have semantic affinity with existing method fragments in OPF. We should avoid introducing redundant new method fragments to OPF. In other words, new formulated method fragments should be checked with existing OPF's method fragments to see if they have counterparts in OPF or not. Therefore, before considering a service-oriented method fragment as a new fragment, we checked if any method fragments existed in the OPF repository covering the new method fragment or not. If not, the method fragment was added to the repository in conformance with the OPEN metamodel standard. Although service-oriented SDMs cover traditional tasks of software development, we have discarded their related method fragments in our presentation in this paper for brevity. For instance, service-oriented SDMs emphasize on business process modeling and business process optimization and OPF supports this emphasis in the business optimization phase.

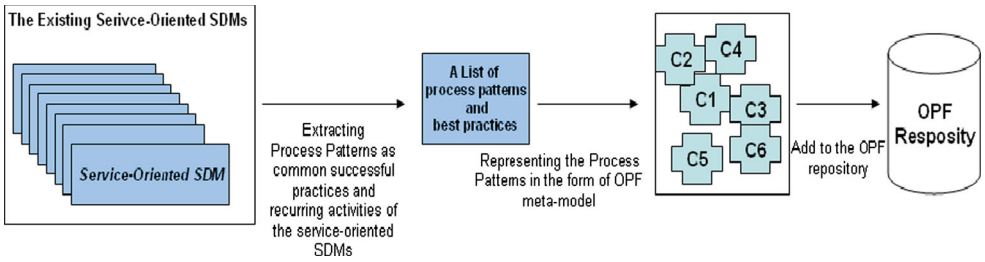

Fig. 3 Existing method re-engineering approach used to obtain new method fragments

We have identified two types of method fragments that are specified in detail in the next section. *Enhanced* ProcessKind method fragments enhance existing ProcessKind method fragments in OPF with new specific service-oriented TaskKind fragments. New ProcessKind method fragments have not been supported by OPF and are new to this framework. Each TaskKind is described in terms of five items as follows [9]: task name, explanation, producer, work products, and supportive techniques and relations. Relations specify relevant predecessor and subsequent of a TaskKind, as well as Deontic matrix that was described in Sect. 2 to denote the relation of a TaskKind with other TaskKinds, ProducerKinds, TechniqueKinds, and WorkProductKinds. Each relation can be mandatory, recommended, or optional.

6 Proposed Service-oriented method fragments

In this section, we elaborate additional method fragments in terms of ProcessKinds and TaskKinds that we propose need to be added to the OPF repository to facilitate service-oriented SDM construction. Each OPF ProcessKind method fragment that is enhanced with new TaskKinds is denoted as an enhanced ProcessKind, while unchanged ProcessKinds are not described in this paper for brevity. Each TaskKind method fragment is presented in terms of task name, a summary of the intent of the task, involved ProducerKind of the task, relevant WorkProductKinds, and supportive TechniqueKinds. For simplicity, sometimes in further sections, we have used the terms process and task for ProcessKind and TaskKind, respectively.

6.1 Enhanced ProcessKind: requirements engineering

In this process, the requirements of the target software system are elicited, specified, and validated by all system stakeholders. This process is very similar to traditional requirements engineering. In fact, it is covered by the existing *requirement engineering* method fragment process in OPF. This is why we consider it as an enhanced process and hence do not explain it again. Only its difference with the *Specify Service Level Agreement (SLA)* task is described. The task is added to the requirement engineering process.

TaskKind Name *Specify Service Level Agreement (SLA)*
Description *Quality of Services* (*QoS*) as a subset of non-functional requirements plays an important role in the service-oriented context. It forces service providers to improve their ability to meet service consumer requirements in a competitive manner with other service providers. Based on the nature of service orientation, various service providers may provide the same service to fulfill consumer's requirements. They can however be different in the QoS they provide. SLA grants the service consumer a degree of guarantee that the service provider complies with and provides acceptable QoS such as security, availability, performance, and reliability in the execution environment. In this task, a contract between service providers and service consumers is established. For instance, it may be contracted that service should respond to input requests only in 20 ms or less.
ProducerKinds Service provider, Service consumer, Requirement engineer
WorkProductKind Document of Service Level Agreement contract
Supportive Technique
Create SLA contract A consensus contracted between service consumer and service provider as *Service Level Specification* (*SLS*) document that specifies a set of typical technical parameters such as the ones listed below:

- *Purpose* The intention of creation of the SLA contract.
- *Parties* The service consumer and provider involved in the SLA and their responsibilities.
- *Validity Time* The period of time that SLA should be met.
- *Scope* The boundaries of and the expectations from SLA.
- *Service-Level Objectives* Level of service quality that service provider and consumer agree on including service availability, security constraints, reliability, latency, and recovery time that are mostly noted in measurable and quantifiable terms.
- *Penalties* Determining what penalties for failure must be paid when SLA contract parties violate the agreements. For example, non-performance may be costly.

Table 2 Possible relation values of Specify Service Level Agreement task

Elements of method fragment	Type of element
Specify Service Level Agreement (SLA)	Task
Service Consumer	Producer
Service Provider	Producer
Requirement Engineer	Producer
Create SLA Contract	Technique
Document of Service Level Agreement Contract	Work product

After SLA is contracted, both the service provider and the service consumer undertake to perform it at runtime in Web service invocations. Table 2 shows Specify Service Level Agreement task method fragment. The third column represents the most recommended values for method fragment.

6.2 Enhanced ProcessKind: environments engineering

This process has many relevant tasks for assessment of the environment, but it is more critical in the context of SOSD. Therefore, in this process, the status of existing infrastructure of enterprise, B2B or systems-of-systems (we refer to as environments) is assessed to find out candidate services from existing assets and to evaluate the readiness and existing capabilities of environment to migrate to service-oriented solution. Moreover, the reasons for migration to service orientation are justified. This process is enhanced with the Evaluate Environment Readiness task.

TaskKind Name *Evaluate Environment Readiness*
Description This task evaluates the readiness of environment to migrate to service-oriented solution. This task contains the following sub-tasks: evaluating the quality of existing codes and software components, evaluating reusability of valuable existing business logics of existing legacy software to expose as Web service, evaluating quality of correctness and integrity of stored data in databases, reconstructing the architecture of existing legacy systems, and providing technology infrastructure and hardware/software resources to support secure, interoperable, and reliable message protocols between services. Even people's attitude towards changes to their environment should be checked to find if it is feasible to build a service-oriented solution or not.
ProducerKinds Requirement engineer, Database administrator, Network administrator
WorkProductKind Report of readiness assessment
Supportive Techniques
Create a readiness report Requirement engineer can perform this task by using well-known criteria of SOA maturity models, such as those proposed by IBM SOAMM and SIMM [68,69]. Table 3 shows the possible relation values of Evaluate Environment Readiness task.

Table 3 Possible relation values of Evaluate Environment Readiness task

Elements of method fragment	Type of element
Evaluate Environment Readiness	Task
Requirement Engineer	Producer
Database Administrator	Producer
Network Administrator	Producer
Create a Readiness Report	Technique
Report of Readiness Assessment	Work product

6.3 Enhanced New ProcessKind: plan project

The aim of this process is to perform preliminary project planning such as scheduling, risk management, and resource planning. This process does not differ from traditional project planning except in plan transition. The process includes one additional task, namely the plan transition task.

TaskKind Name *Plan Transition*
Description This task is performed to adopt various strategies based on situations of the environment and the state of existing legacy systems (software components) for transition to service orientation [70]:

- *Replacement Strategy* In this strategy, existing legacy systems are retired entirely by rewriting them from scratch and constructing a new service-oriented system. Although this strategy can be expensive and time consuming, it can lead to a solution that fits better to the requirements of the service consumer.
- *Wrapping Strategy* Some parts of the existing valuable business logics of legacy systems are wrapped by Web service technology (e.g., .Net or J2EE), and then exposed as a service to consumers.
- *Redevelopment Strategy* This strategy uses the re-engineering approach to add new services to existing legacy systems.
- *Migration Strategy* This strategy incorporates both redevelopment and wrapping, and aims to develop a new system with an improved service-oriented solution.

Having selected the strategies, a transition plan that preserves functionalities of the original system for migration to service orientation is developed. Several strategies may be pursued at the same time based on the situation of existing systems. The task finishes with a primary estimation effort, cost, and definition of a road map for migration to service orientation. It should be noted that the plan can be updated any time.
ProducerKind Service consumer, Service provider, Project manager

Table 4 Possible relation values of Plan Transition task

Elements of method fragment	Type of element
Plan Transition	Task
Service Consumer	Producer
Service Provider	Producer
Project Manager	Producer
Make Transition Plan	Technique
Transition Plan	Work product
List of Transition Issues	Work product
Cost and Effort of Selected Strategies	Work product

WorkProductKinds Transition plan, List of transition issues, Cost and effort of selected strategies

Supportive Techniques

Make Transition Plan The purpose of the proposed technique is to make a document in which possible alternatives for migration to service orientation are clarified, discussed, justified, and critical milestones scheduled and documented. Table 4 shows the possible relation values of Plan Transition task.

6.4 New ProcessKind: develop SOA governance model

In this new process, a governance model is established and then cuts through all development process. Because service orientation involves various service providers and consumers that may work in a geographically distributed environment, a governance model should be established to ensure that the adoption of service orientation are constantly aligned with IT initiatives and business needs. Indeed, the process acts as an umbrella process over software development that is performed continuously. This new process includes only one task.

TaskKind Name *Develop Governance Model for Current Iteration*

Description Service consumers and providers collaborate to establish chains of responsibility, authority, communication, and overall scope as well as solution size and funding for performing the governance model in the current iteration of the solution. Details of governance model mainly include a set of supportive high-level policies and rules to achieve right services that essentially relate to QoS. Executive mechanisms are defined to realize the defined policies. Finally, the task defines as much as possible quantifiable metrics and indicators to measure and monitor QoS during service usage.

ProducerKind Service consumer, Service provider, Requirements engineer

WorkProductKinds Documented (textural description) governance model, policies, executive mechanisms, quality indicators and measurement metrics.

Table 5 Possible relation values of Develop Governance Model for the Current Iteration task

Elements of method fragment	Type of element
Develop SOA Governance Model	Process
Develop Governance Model for Current Iteration	Task
Service Consumer	Producer
Service Provider	Producer
Requirements Engineer	Producer
Create Governance Model	Technique
Documented (Textural Description) Governance Model	Work product
Policies	Work product
Executive Mechanisms	Work product
Quality Indicators and Measurement Metrics	Work product

Supportive Techniques

Create Governance Model There are many techniques that service consumer and provider can accommodate as a governance model to develop service-oriented software successfully, such as the one proposed by IBM [71]. Table 5 shows the possible relation values of Develop Governance Model for Current Iteration task.

6.5 New ProcessKind: Design Services

The Design Services process is the core of SOSD. When the main business processes are identified and re-engineered, useful services that encapsulate business logic capabilities are defined. This process takes a set of business process models as input and yields a set of candidate services as output. The process has four tasks.

TaskKind Name *Identify Services*

Description In this task, existing business processes and subprocesses are translated (manually, semi-automatically or fully automatically) into one or more services to be exposed to business partners. In other words, valuable services aligned with IT initiatives are identified. This results in a blueprint (big picture) of service-oriented environment [63]. Definitions of identified services are more high level and abstract than the specific details of the service operations that are specified rigorously later in the *Specify Details of Services* task.

ProducerKind Service designer (service modeler) as a member of service provider

WorkProductKinds Service models, Services interfaces signatures

Supportive Techniques There are three well-known techniques (typically referred to as strategy) for service identifica-

Table 6 Possible relation values of Identify Services task

Elements of method fragment	Type of element
Design Services	Process
Identify Services	Task
Service Designer	Producer
Top-Down	Technique
Bottom-Up	Technique
Meet-In-The Middle	Technique
Service Models	Work product
Services Interfaces Signatures	Work product

tion, namely [52,58]: top-down, bottom-up and meet-in-the middle (agile). In the top-down technique, a preliminary set of service interfaces become candidate and grouped into a logical context and further elaborated in the Specify Details of the Services task. Specifically, the technique focuses on identifying candidate services such as business services from the environment of business process models. The steps of business processes are transformed to a set of candidate services. The bottom-up technique concentrates on wrapping the underlying existing legacy logics into services that are built on top of legacy systems to make them easily accessible to other systems. This technique redirects the environment to new ways of supporting business needs. The agile technique proposes a combination of top-down and bottom-up techniques. Services can be modeled and presented by UML 2.0 profile for SOSD [72]. Table 6 shows the possible relation values of Identify Services task.

TaskKind Name *Specify Details of Services*
Description The definition of defined services are consolidated with more specific details such as interface specification, service dependencies, operation signatures, operation parameters and parameter types, and input/output messages.
ProducerKind Service designer (service modeler)
WorkKindProducts Service interfaces specifications, Realizer components, Service dependencies
Supportive Techniques
Add Specific Details to Services Service designer refines candidate services. They design interfaces to provide interoperability between service providers and consumers, input and output parameters, and error messages for services operations. Operations of services are detailed via analyzing collaborations between services. Instantiation of UML 2.0 class, interface, and collaboration stereotypes [72] are used to represent services specifications. Service designer looks for potential software components that can realize service functionalities. Table 7 shows the possible relation values of Specify Details of Services task.

TaskKind Name *Classify Services*
Description In this task, various types of identified services

Table 7 Possible relation values of Specify Details of Services task

Elements of method fragment	Type of element
Design Services	Process
Specify Details of Services	Task
Service Designer	Producer
Add Specific Details to Service	Technique
Service Interfaces Signatures	Work product
Software Components Specification	Work product
Service Dependency	Work product

Table 8 Possible relation values of Classify Services task

Elements of method fragment	Type of element
Design Services	Process
Classify Services	Task
Service Designer	Producer
Classify Service	Technique
Classified Service Model	Work product

are classified based on the usage context. The most well-known manageable classification for services is typically hierarchical, in which services are classified based on the degree of granularity from coarse-grain to fine-grain services, e.g., mission-aligned business services, enterprise services, application services and utility (also named infrastructure) services. The intent of performing the task is to facilitate clear, precise, and non-overlapping definitions for the wide range of services in the environment and may be used during a service-orientation initiative. The classification assists service providers (developers) to have more effective communication with service consumers, to understand their state of existing assets, and to derive a blueprint for the service-oriented environment.
ProducerKind Service designer
WorkProductsKind Classified service models presented by UML 2.0 stereotypes for service classification
Supportive Techniques
Classify Service This technique is performed to classify services based on their objectives and characteristics, e.g., business services, application services, and utility services. The classification helps service providers and service consumers to identify which services will be used in the SOA layers [63]. Table 8 shows the possible relation values of Classify Services task.

TaskKind Name *Evaluate Quality of Designed Services*
Description The aim of this task is to increase maintenance, simplicity, changeability, future enhancements, and reuse of services. More precisely, the design quality of services is evaluated in terms of *Granularity, Coupling, Cohesion,* and

support of *Reusability*. The number or scope of functionalities of a service is named service granularity and is identified as a coarse-grain or a fine-grain service [51]. The appropriate level of service granularity has direct effect on service coupling and cohesion. Evaluating the coupling of services is performed to minimize dependency (e.g., data dependency and resource dependency) between services. While business processes are realized via orchestration of services, the dependency between services should be as low as possible to provide more agility of business processes, while underlying business processes and rules change more frequently upon business needs. Low coupling increases service reusability for future projects. Evaluation of cohesion is performed to check whether a service exposes a set of relatedness of necessary functionalities or not. It should be noted that a trade-off is needed while taking into account granularity, coupling, cohesion, and service reusability.

ProducerKind Service desginer

WorkProductKinds Refined service model

Supportive Techniques There are three specific service-oriented techniques for performing this task.

Evaluate Service Granularity Service granularity can be evaluated in different ways such as by the number of software component interfaces invoked for a given service operation [64]. When service operations increase, the sizes of messages and data transfers increase and create higher dependency on the context. In contrast, fine-grain services increase the number of message passing between them.

Evaluate Service Coupling The Service designer utilizes techniques such as the one proposed by Perepletchikov et al. [73], in which a suite of 17 quantified service-coupling metrics are used to measure service coupling. Based on the evaluation results, the service modeler may revise the service model. Prescriptive guidelines can be incorporated during service design too [46].

Evaluate Service Cohesion The Service designer can use this technique to determine if the functionalities of a designed service are cohesive, for example if coincidentally and sequentially of operations of Web Services are satisfied or not [74]. Table 9 shows the possible relation values of Evaluate Quality of Designed Services task.

Table 9 Possible relation values of Evaluate Quality of Designed Services task

Elements of method fragment	Type of element
Design Services	Process
Evaluate Quality of Designed Services	Task
Service Designer	Producer
Evaluate Service Granularity	Technique
Evaluate Service Coupling	Technique
Evaluate Service Cohesion	Technique
Refined Service Model	Work product

6.6 Enhanced ProcessKind: Service-Oriented Architecture Engineering

This process is supported by existing *Architecture Engineering* process in the OPF repository that we renamed it to *Service-Oriented Architecture Engineering* to promote it to SOA. The main enhancement relates to instantiation of stack-based service-oriented reference architecture (SOA reference model) [52] in which services are organized into different layers. The layered architecture enables complexity management and facilitates the decision to where to place services and how to provide support for SOA-specific QoS issues. QoS is realized by utilizing well-known architecture strategies and tactics such as the ones proposed in [75].

6.7 Enhanced ProcessKind: develop services

This process enhances the *Implementation* process of the OPF repository. The real required services such as business services, enterprise services, application services and utility are developed in various manners. The process includes three tasks as follows.

TaskKind Name *Implement and Test Necessary Services*

Description If no suitable required Web service is found in OPF or no exact match with the requirements is found, an alternative implementation must be developed from scratch. Services are implemented and tested by service provider (development team). Meanwhile, specification of the implemented service as a Web service is expressed in Web Service Description Language (WSDL), wherein public available operations are exposed in a way that service consumers can invoke them. Because Web Services can be developed separately by a geographically distributed development team, all Web Services as part of a distributed system should be tested independently and integrated with other Web Services or systems involved. Test performed by service provider and service consumer. Service provider can provide a number of test cases for service consumer to reuse.

ProducerKind Service developer, Service tester

WorkProductKinds Executable Web Services, Services WSDLs and WS-Policy

Supportive Technique

Implement Services Service developer uses this technique. Existing OO analysis and design techniques such as analyzing and designing classes, CRC card modeling and classifying relevant classes into cohesive software components are used to implement services. From an implementation viewpoint, a Web service realizes a service comprising a number of software components. Specifications of software components provide the basis for the design and implemen-

Table 10 Possible relation values of Implement and Test Necessary Services task

Elements of method fragment	Type of element
Implement and Test Necessary Services	Task
Service Developer	Producer
Service Tester	Producer
Implement Services	Technique
Perform WSDL Testing	Technique
Executable Web Services	Work product
Services WSDLs and WS-Policy	Work product

tation of Web Services, i.e., service interfaces. OPEN has a set of method fragments that allows for incorporating CBD approach in the software development process. The Implement Services task forces service providers to accommodate existing tasks of OPF that are specified in the Implementation process.

Perform WSDL Testing In addition to traditional testing techniques, Service tester can use the WSDL testing technique. Web Services have WSDL as the only available interface at testing time. WSDL metadata files are XML documents containing information about Web service's operations and required QoSs. Test case generator tools use WSDL files to generate test cases automatically. Test cases act as SOAP messages sent to Web Services as well as to service consumers. All Web service operations include various inputs/outputs with different data types. Confidentiality and integrity of SOAP messages during test should be taken into account too. Table 10 shows the possible relation values of Implement and Test Necessary Services task.

TaskKind Name *Implement Necessary Wrappers*

Description This task concentrates on the software components comprising the interfaces of existing legacy systems. Based on the work products of the Evaluate Environment Readiness task, valuable business logics of one or more existing legacy systems that provide desired functionalities are extracted and exposed through universal standard Web service technologies such as .Net or J2EE. Wrapping provides new broad accessibility Web service interfaces to existing legacy software components. Wrapping existing software components interfaces as Web Services is justifiable when the development of service-oriented systems from scratch is expensive, risky, and time consuming.

ProducerKind Service developer

WorkProductKinds Executable Web Services, Services WSDLs

Supportive Techniques

Implement Wrappers There are several step-by-step techniques including manual [76,77], semi-automatically [78,79], or fully automatically [80] techniques that service developers can use to wrap individual functionalities in legacy source codes such as Web Services. Table 11 shows the possible relation values of Implement Necessary Wrappers task.

Table 11 Possible relation values of Implement Necessary Wrappers task

Elements of method fragment	Type of element
Implement Necessary Wrappers	Task
Service Developer	Producer
Service Tester	Producer
Implement Wrapper	Technique
Executable Web Services	Work product
Services WSDLs	Work product

TaskKind Name *Develop Necessary Composite Web Services*

Description This task is composed of a number of prepared fine-grain (also called atomic) Web Services and other software components related to the underlying business processes that form a more coarse-grain Web service called a Composite Web Service that is assumed to maximize business value. In fact, service consumers synthesize Composite Web Services to realize ultra large-scale software system in terms of systems-of-systems or supply chain management via composing a dozen of heterogeneous distributed independent Web Services. Definition of business service is adopted from Business Process Modeling Language (BPML) [81] and IBM's WSFL [82] wherein the invocation order of Web Services—orchestration or choreography—and the sequencing of message passing and bindings between services are defined to form the flow of business services. It is worth noting that the composition of appropriate Web Services with guaranteed QoS should be considered prior to the construction of a service-oriented system. Therefore, analysis of Web Service composition alternatives to select the best composition must be done. Composite Web Services are executed later by Business Process Execution Language for Web Services (BPEL4WS) [83], which is a standard engine for business process execution. The task is conducted using manual, semi-automated, or automated composition techniques [84].

ProducerKind Service consumer, Business process engineer

WorkProductKinds Composite services as business process (BPEL processes)

Supportive Techniques

Compose Web Service There are several supportive techniques for this task [85]. A service consumer takes a number of fine-grain Web Services to configure a given business process model. Then he/she evaluates how best the composed Web Services meet the desired functionalities and QoS parameters to select the best composition. Service composition task becomes more complex as the number of provided Web Services (e.g., available Web Services on the

Table 12 Possible relation values of Develop Necessary Composite Web Services task

Elements of method fragment	Type of element
Develop Necessary Composite Web Services	Task
Service Consumer	Producer
Business Process Engineer	Producer
Compose Web Services	Technique
Composite Services as Business Process	Work product

Web) increases. Therefore, automated or semi-automated tools to help service consumers in this hard task are critically required. Table 12 shows the possible relation values of Develop Necessary Composite Web Services task.

6.8 Enhanced ProcessKind: Reuse Engineering

We have only enhanced this existing OPF process with one service-oriented specific task.

TaskKind Name *Discover Necessary Web Services*
Description The aim of this task is to help search and select from existing Web Services, those that match best with the service consumers' requirements such as QoS and functionalities. The result of this task is a list of retrieved candidate Web Services. Given that Web Services can be developed by various service providers, services should be certified to ensure that selected services satisfy the required quality of concerns (SLA). It is possible that many Web Services exactly match the particular requirements. Therefore, the service consumer must evaluate them all to select the best ones. For paid services, a usage-based billing model for charging of services is contracted between service provider and service consumer. Typically, the discovery task is supported by automatic Web service discovery engines.
ProducerKind Service consumer
WorkProductKinds Executable Web Services
Supportive Techniques
Search Web Services Service consumers can use generic search engines such as Google to find WSDL documents in the Web or running SOAP APIs that allow performing queries on UDDI directories. Tools can assist service consumers to locate services that accurately satisfy the required QoSs. Table 13 shows the possible relation values of Discover Necessary Web Services task.

6.9 Enhanced ProcessKind: enable service-oriented solution

This process is mainly supported by *Deployment* process in OPF. The *Service-Oriented Solution* enhances this process by two service-oriented specific tasks. In this process, Web Services are deployed in an operational environment. Moreover, defects and missing requirements are discovered in this process. In some cases, it is difficult to determine a time for deployment of Web Services as building blocks of the system when a service-oriented system can be fully developed via existing Web Services that have already been provided and published by various service providers.

Table 13 Possible relation values of Discover Necessary Web Services task

Elements of method fragment	Type of element
Discover Necessary Web Services	Task
Service Consumer	Producer
Search Web Services	Technique
Executable Web Services	Work Product

TaskKind Name *Publish Web Services*
DescriptionWeb Services are hosted and advertised by service providers and published to an accessible common registry such as a Universal Description Discovery and Integration (UDDI) server (http://www.uddi.org/). The major information in addition to what is provided for a typical Web Service includes Web Service's operations signatures and QoS values such as the cost of usage, availability, and security issues. Service consumer can discover the required Web Services through universal protocols such as SOAP messages. In fact, service providers advertise their Web Services at a global marketplace on the Web.
ProducerKind Service installer
WorkProductKinds Deployed and published services
Supportive Techniques
Import Web Services into a Common Web Service Repository The service installer takes tested Web Services, generates a Web Service description document like WSDL for each one, and publishes it to a common repository such as in a UDDI where service consumers can find Web Services. Table 14 shows the possible relation values of Publish Web Services task.

TaskKind Name *Perform Test in Large*
Description This task tests orchestrated or choreographed

Table 14 Possible relation values of Publish Web Services task

Elements of method fragment	Type of element
Publish Web Services	Task
Service Installer	Producer
Import Web Services into the Common Web Service Repository	Technique
Deployed and Published Services	Work Product

Table 15 Possible relation values of Perform Test in Large task

Elements of method fragment	Type of element
Perform Test in Large	Task
Orchestrator/Choreographer Tester	Producer
Perform Orchestration/Choreography Testing	Technique
Test Cases	Work Product
Results of Running Test Cases	Work Product

Web Services to see if the composition of Web Services that builds a distributed system actually meets the business acceptance criteria for functional requirements and SLA for non-functional concerns. Based on the nature of SOSD, such a test typically involves more than one software development team (service provider) and business partner (service consumer), such as when a composite Web Service realizes a business-to-business (B2B) business process.
ProducerKind Orchestrator/Choreographer Tester
WorkProductKinds Test cases, Result of running test cases
Supportive Techniques
Perform Orchestration/Choreography Testing One way to perform this task is to define certain business process scenarios as test cases. The results of performing the tests are compared with expected functionalities, SLA contracts, specially predefined policies, and quality criteria in the SOA governance criteria. Table 15 shows the possible relation values of Perform Test in Large task.

6.10 Enhanced ProcessKind: maintenance

After Web Services are fully deployed in an operational environment, this process evaluates QoS of all participating Web Services that make up the distributed system, against predefined SLA contract and SOA governance model continuously. This process includes one main task.

TaskKind Name *Monitor Operational Web Services*
Description The aim of this task is to indicate service degradation, noncompliance with service-level offerings, and service availability levels before service failure actually occur. To do this, service consumers gather and log data during Web Services usage. They then measure and interpret Web Services against predefined metrics in Develop SOA Governance model and the SLA contract. For Web Services having usage-based billing models as well as those consensued in the SLA contract, service proivders generate billing reports to service consumers to pay them.
ProducerKind Service consumer, Service provider
WorkProductKinds Statically generated reports of QoS, Service metering, Billing report and Defect report.
Supportive Techniques

Table 16 Possible relation values of Monitor Operational Web Services task

Elements of method fragment	Type of element
Monitor Operational Web Services	Task
Service Consumer	Producer
Service Provider	Producer
Monitor QoS of Web Services	Technique
Statically Reports of QoS	Work Product
Service Metering	Work Product
Billing Report and Defect Report	Work Product

Monitor QoS of Web Services Service consumers log and analyze Web Service invocations, for instance the number of Web Service operation invocations or the number of authentication failures, to detect violations from promised QoS parameters such as response time, throughput, and availability. Historical information of Web Services' QoSs are analyzed. Based on the generated reports, business processes may need management decisions to accommodate changes to business processes (as composite Web Services) such as Web Service replacement with another one for continuous QoS improvement (See the *Compose Web Service Dynamically* task). Table 16 shows the possible relation values of Monitor Operational Web Services task.

TaskKind Name *Compose Web Services Dynamically*
Description This task allows utilizing various Web Services on the Web on demand without enforcing any Web Service composition or deployment in advance. This way, service composition that is usually performed at design time can be done dynamically at runtime too. Consequently, this task blurs the distinction between tasks at design time and runtime. Malfunctioning of Web Services at runtime in the system can be sensed and rectified dynamically by probing for new Web Services and replacing them with new ones on the fly.
ProducerKind Service consumer
WorkProductKinds New Discovered Web Services
Supportive Techniques
Reconfigure Composite Web Services Service consumers reconfigure composite Web Services in which degraded Web Services have been detected and replaced by new ones. Typically, dynamic Web Service composition is performed automatically. Table 17 shows the possible relation values of Compose Web Service Dynamically task.

Relation among method fragments Although method fragments are stored independently in the repository, the constraints on them can be specified via Constraint superclass of OPEN [14,15]. Constraint superclass provides a linkage as well as a predecessor and a subsequent one for method fragments. There are two subtypes of Pre-Condition and

Table 17 Possible relation values of Compose Web Service Dynamically task

Elements of method fragment	Type of element
Compose Web Services Dynamically	Task
Service Consumer	Producer
Reconfigure Composite Web Services	Producer
New Discovered Web Services	Technique

Post-Condition for this purpose that we have used. The constraints allow us to clarify imperative constraints on method fragments. Figure 4 shows possible predecessor and subsequent constraints as pre-conditions and post-conditions of the use of task method fragments as recommended in OPEN. The directions of arrows show dependency between tasks For instance, Identify Services task should be completed for identifying a list of required services before performing the Discover Necessary Web Services for obtaining executable Web Services. It is obvious that all constraints are maintained as Pre-Condition and Post-Condition fields of the task method fragments.

7 Position of new method fragments in OPEN

Our proposed method fragments represent necessary service-oriented method fragments that should be added to the OPF repository in support of service-oriented SDM construction using OPF method fragments. To construct a project-specific service-oriented SDM, required process should be selected first. Then, task method fragments should be selected to complete the internal details of the process of method fragments. For each task method fragment, relevant producer(s), work product(s), and supportive technique(s) should be determined.

Table 18 shows the position of the new service-oriented method fragments in the OPEN process model as an enhancement to the OPF repository in order to incorporate service-oriented method fragments. The original process method fragments (first column) of OPEN with ten processes form the OPEN development process model. New process and task method fragments enhance OPEN in portions that service-oriented support is needed. The two new processes of method fragments are Design Services and Develop Governance. Each of these new processes has new task method fragments themselves. The original processes of method fragments are extended with new service-oriented task method fragments (second column). For instance, the Requirements Engineering process is enhanced with Specify SLA task method fragment. For brevity, task method fragments originally existing in OPF are not shown here (for details see [14,15]). The third column shows the Producer that should be employed to produce necessary Work Products (forth column). Tasks are performed to complete the processes. Supportive techniques should be used to realize tasks. For instance, the Design Services process has four associated tasks.

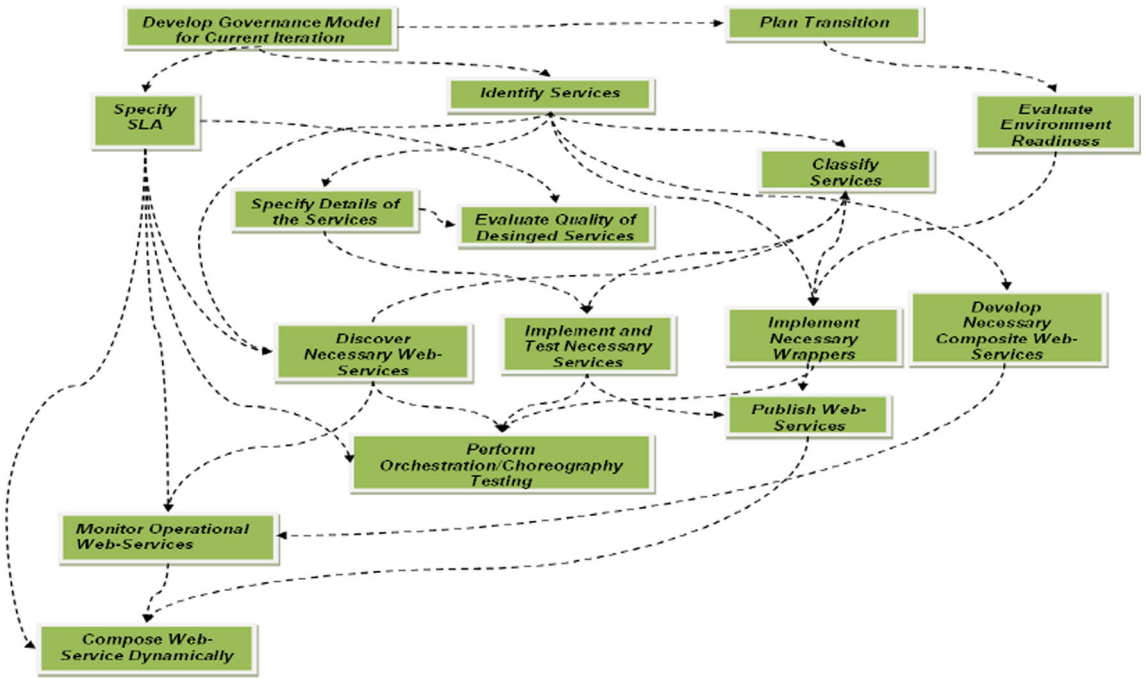

Fig. 4 Relation among method fragments

Table 18 New service-oriented specific method fragments incorporated into OPEN

ProcessKind	TaskKind	TechniqueKind	WorkProductKind	ProducerKind
Requirements Engineering	Specify SLA	Create SLA Contract	Document of Service Level Agreement Contract	Service Provider, Service Consumer, Requirement Engineer
Environments Engineering	Evaluate Environment Readiness	Evaluate Environment with SOA Maturity Model Criteria	Report of Readiness Assessment	Requirement Engineer, Database Administrator, Network Administrator
Develop Governance	Develop Governance for Current Iteration	Create Governance Model	Documented Governance Model, Policies, Executive Mechanisms, Quality Indicators and Measurement Metrics	Service Consumer, Service Provider, Requirements Engineer
Reuse Engineering	Discover Necessary Web Services	Search Web Services	Executable Web Services	Service Consumer
Design Services	Identify Services	Top-Down Analysis Bottom-Up Analysis Meet-In-the Middle Analysis	Service Models Service Interface Signatures	Service Designer
	Specify Details of Services	Add Specific Details to the Service	Service Interface Signatures Realizer Components Service Dependency	Service Designer
	Classify Services	Classify Service	Classified Service Model	Service Designer
	Evaluate Quality of Designed Services	Evaluate Service Granularity Evaluate Service Coupling Evaluate Service Cohesion	Refined Service Model	Service designer
Implementation	Implement and Test Necessary Services	Implement Services Perform WSDL Testing	Executable Web Services	Service Developer Service Tester
	Implement Necessary Wrappers	Implement Wrappers	Services WSDLs and WS-Policy	Service Developer
	Develop Necessary Composite Web Services	Compose Web Service	Executable Web Services Services WSDLs Composite Services as Business Process	Service Consumer Business Process Engineer
Deployment	Publish Web Services	Import Web Services into the common Web Service Repository	Deployed and Published Services	Service Installer
	Perform Test in Large	Perform Orchestration or Choreography Testing	Test cases Result of running Test Cases	Orchestrator/Choreographer Tester
Maintenance	Monitor Operational Web Services	Monitor the QoS of Web Services	Static Reports of QoS Service Metering Billing Report and Defect Report	Service Consumer Service Provider Service Consumer
	Compose Web Service Dynamically	Reconfigure Composite Web Services	New Discovered Web Services	Service Consumer
Management	Plan Transition	Make Transition Plan	Transition Plan List of Migration Issues Cost and Effort of Selected Strategies	Service Consumer, Service Provider, Project Manager

8 Evaluation

We presented a set of reusable service-oriented method fragments to facilitate the construction of situational SDM methods based on situational factors of the project at hand. Whenever an organization aims to construct a service-oriented SDM, it can construct its SDM based on our proposed set. The question is though how much valid and correct are these fragments? We thus need to verify and validate (V&V) our proposed method fragments. To do this, we need to state clearly, what we mean by V&V of a set of method fragments. Unfortunately, there is no pertinent and approved definition or analysis criteria to verify and validate a set of method fragments to be added to the OPF repository.

Henderson-Sellers and Gonzalez have conducted a theoretical work on the granularity and the size of the resulting method fragments [88]. They have argued that granularity affects reusability of method fragments and thus method fragments should be atomic rather than being coarse grained. However, their work is in progress and not finalized yet. Therefore, we could not find a mature metric or evaluation criteria to analyze our method fragments in detail. Furthermore, most existing evaluation criteria, which often use qualitative questionnaires, have focused on evaluating the quality of the constructed situation SDM rather than the method fragments themselves [19].

For the above reasons, we decided to use the abstract definition of V&V about software artifacts proposed by Bohem's [89] and Pressman [90]:

1. *Abstract Verification* Has the artifact been constructed in the right way?
2. *Abstract Validation* Has the right artifact been constructed?

Using these definitions, we made an analogy between the terminologies of V&V in the realms of software engineering and SME, specifically method fragments. To be more specific, we have concretized V&V for SME as follows:

1. *Concrete Verification* Has the proposed method fragments been constructed/identified in a right manner in line with the OPEN/OPF and SME objectives?
2. *Concrete Validation* Has the right method fragments been developed to facilitate the construction of various SDMs?

We argue that we have defined the proposed method fragments in the right manner (i.e., are verified) based on the first definition. This is because we have used the method of re-engineering approach proposed by Ralyté [7] and systematically reviewed the main sources and published literature on OPEN metamodel [9,14], OPF repository, and construction of method fragments [19]. We then extracted the recurring fragments from 11 prominent service-oriented SDMs to ensure the resulting method fragments were [16] non-redundant, without overlapping, and compatible with the structure of the latest revision of OPEN's metamodel, namely the ISO/IEC 24744 [25]. We made sure to represent the proposed method fragments in the same structure as that of ISO/IEC 24744 to make them consistent with the method fragments already stored in OPF and thus easily connectable to preexisting method fragments.

To validate that we have developed the right method fragments, we use two criteria presented in [91–94], namely *usability* and *completeness*. The usability criterion measures the range of situational SDMs that can be constructed from the proposed method fragments based on the projects' requirements. The completeness criterion measures how fully the proposed method fragments cover any specific domain of software development. In the following two sections A and B, we separately argue that the proposed method fragments satisfy these two criteria in practice.

8.1 Completeness

We use *Domain Fragment*s and *Domain Coverage* presented by Han [94] to validate the completeness of our proposed method fragments. Domain coverage measures the adequacy of a set of proposed method fragments in covering a specific domain of software development, while domain fragments are a subset of a domain and propitious domain fragments are those that more fully cover that domain. We thus need to define a domain for validating the completeness of our proposed method fragments first. Considering the Papazoglou's recommendation [2] that argues in favor of service-oriented SDMs as a suitable representative domain for service-oriented paradigm or service-oriented software engineering/computing, service-oriented SDMs constitute the domain of our work. Fortunately, we had in fact selected this domain before to derive the proposed method fragments.

As stated in Sect. 4 before, we had selected a number of prominent service-oriented SDMs based on their applications in real projects, their maturity levels, their citation rates, accessibility to their resources, and quality of their documentations [49]. The following 11 service-oriented SDMs were chosen: IBM SOMA, SUN SOA Repeatable Quality (RQ), CBDI-SAE Process, MSOAM, IBM RUP for SOA, SDM proposed by Papazoglou, IBM SOAD (Service-Oriented Analysis Development), Service-Oriented Unified Process (SOUP), SDM proposed by Chang and Kim, Steve Jones' Service Architectures, Service-Oriented Architecture Framework (SOAF). Therefore, all these 11 SDMs constitute the domain of our work and the set of proposed method fragments constitute the domain fragments. We should thus show that the proposed method fragments *(i.e., domain fragments)* cover these 11 service-oriented SDMs *(i.e., domain)* adequately. We define two general equations below (Eqs. 1, 2) to measure the adequacy of this coverage, wherein

- *Task* refers to a substantial task in an SDM. It is a bit inductive and tentative to figure out which elements in an SDM are tasks. Some examples include *Requirement elicitation, Design prototypes, Evaluate software architecture*, and *Implement code*.
- *Number of Tasks (NT)* represents the total number of tasks in an SDM.
- *Method Fragment (MF)* represents a typical method fragment.

- *Sum of Method Fragments (SMF)* represents the total number of service-oriented task method fragments, which is 16 in our case here in this paper.
- *N* represents the number of SDMs, which is 11 in our case here.
- *Method Coverage (MC)* represents the degree of coverage of a service-oriented SDM (a SDM is a subset of domain) by a set of service-oriented method fragments (domain fragments) that is calculated by Eq. 1.
- *Domain Coverage (DC)* represents the degree of coverage of the service-oriented SDMs (domain) by service-oriented method fragments (domain fragments) that is calculated by Eq. 2.

$$\mathrm{MC} = \frac{\sum_{i=1}^{\mathrm{SMF}} \mathrm{MF}_i}{\sum_{i=1}^{\mathrm{NT}} \mathrm{Task}_i} \quad \begin{cases} > 1 \\ = 1 \\ < 1 \end{cases} \tag{1}$$

$$\mathrm{DC} = \begin{cases} 1 & \forall\, \mathrm{MC} \in \mathrm{domain} \Rightarrow \mathrm{MC} = 1 \\ 0 & \text{else} \end{cases} \tag{2}$$

An MC $>$ 1 means that the proposed service-oriented method fragments not only cover an SDM, but that they provide more tasks than required by the SDM. In other words, the SDM can be constructed by reusing the proposed method fragments. An MC = 1 implies a one-to-one relation between method fragments and domain. An MC $<$ 1 means that method fragments are not adequate to cover the SDM fully and that they should be enriched with more method fragments. DC is one when method fragments cover all SDMs, and is zero when they fall short of covering all SDMs.

Table 19 shows the calculated MC values for each SDM by the proposed method fragments, using Eq. 1. It should be noted that the calculation of the number of tasks in SDMs was difficult because tasks were represented mostly in a narrative form rather than in a formal format. We thus used the process-centered textual template proposed by Ramsin [95] to categorize tasks and facilitate their enumerations. Therefore, the second column of Table 19 shows the list of decomposed tasks of each SDM using this process-centered template. For brevity, we did not consider traditional tasks of SDMs in this template. For example, in the IBM RUP for SOA, there were three main service-oriented tasks. The third column demonstrates the correspondence between one or more proposed service-oriented task method fragments to each task of each SDM. In other words, the second and third columns together show a one-to-one mapping between the tasks of SDMs and the proposed set of method fragments.

As shown in Table 19, the MC values for all 11 SDMs were lower than 1. For example, given our 16 proposed method fragments (SSMF = 16), IBM SOAD with three tasks (NA = 3) had an MC equal to 3/16 (0.1870) indicating that the proposed method fragments not only cover this SDM, but provide more support than it requires. In other words, a method engineer can construct IBM SOAD with the proposed task method fragments. The same is true for other SDMs too.

Given that the MC values for all 11 SDMs were lower than 1, the DC value is 1 indicating that the proposed task method fragments cover *all* of SDMs, or that all these SDMs can be constructed using the proposed method fragments. We have thus shown that the proposed method fragments *(i.e., domain fragments)* cover these 11 service-oriented SDMs *(i.e., domain)* adequately, or better said are complete only in this context.

8.1.1 Gap analysis

As far as the completeness of the proposed method fragments derived from our study of the 11 prominent service-oriented SDMs is concerned, it should be noted that the proposed set of fragments, as a core for the construction of service-oriented SDMs, may be enhanced further and evolved by the introduction and consideration of any new service-oriented SDMs. The analysis of new service-oriented SDMs can lead to the addition of a new assortment of method fragments too. However, as more and more new SDMs are considered, we expect that the incremental additions to the proposed method fragments diminish marginally. The same argument applies to any other existing service-oriented SDM we had not considered in our research such as the Multi-View SOAD proposed by Kenzi et al. [96]. We only claim and show that the proposed method fragments are complete with respect to the 11 selected prominent service-oriented SDMs.

8.2 Usability

Having shown the *completeness* of the proposed method fragments in Sect. VII-A, we must now show that the proposed fragments are *usable* in the construction of situational SDMs based on situational factors of the project at hand. These two properties together validate our proposed method fragments.

A real empirical assessment is required to justify the usability property of the proposed fragments. However, we have two reservations. Firstly, "Software Process Assessment" is still considered as a challenging task in the SME literature [19,39] and few real case studies can be found to indicate industrial usages [86]. Secondly, performing a wide-range of empirical experiments on the usability of the proposed service-oriented method fragments in several industrial projects and in different software development organizational circumstances would seem to be an ideal way to evaluate our work. However, adopting such an evaluation technique requires considerable amount of time, effort, and resourses to monitor, gather, and measure data continuously

Table 19 Coverage of eleven service-oriented SDMs by the proposed service-oriented method fragments

SDM	Task	Corresponding task method fragment(s)
IBM SOAD	1. Service Identification	Identify Services
	2. Service Classification	Classify Services
	3. Service Modeling and Documentation	Specify Detail of Services
	NA = 3; SSMF = 16; MC = 3/16 (0.187)	
IBM SOMA 2008	1. Business Modeling and Transformation	Business Requirements Engineering (from Requirements Engineering process method fragment in OPF)
	2. Solution Management	All tasks in Project management Process method fragments in OPF
	3. Identification	Identify Services
	4. Specification	Specify Detail of Services
	5. Realization	Candidate Component Evaluation and
		Candidate Component Solution Identification in OPF (from Component Product Acquisition process method fragment in OPF repository)
	6. Implementation	Implement and Test Necessary Services
		Implement Necessary Wrappers
	7. Deployment, Monitoring, and Management	Publish Web Services
		Monitor Operational Web Services
		Compose Web Services Dynamically
	NA = 7; SSMF = 16; MC = 7/16 (0.437)	
CBDI-SAE Process	1. Manage	Evaluate Environment Readiness
		Develop Governance Model for Current Iteration
	2. Consume	Business Requirements Engineering (from Requirements Engineering process method fragment in OPF)
	3. Provide	Plan Transition
		Service-Oriented Architecture Engineering
		Implement and Test Necessary Services
		Implement Necessary Wrappers
	4. Enable	Publish Web Services
		Monitor Operational Web Services
		Compose Web Services Dynamically
	NA = 4; SSMF = 16; MC = 4/16 (0.25)	
SOUP	1. Incept	Evaluate Environment Readiness and Business Requirements Engineering (from Requirements Engineering process method fragment in OPF)
	2. Define	Plan Transition
		Identify Services
		All tasks in Project management Process method fragments in OPF
	3. Design	Specify Detail of Services
	4. Construct	Implement and Test Necessary Services, Implement Necessary Wrappers
	5. Deploy	Publish Web Services
	6. Support	Monitor Operational Web Services
	SMA = 6; SSMF = 16; MC = 6/16 (0.375)	
MSOAM	1. Service-Oriented Analysis	Evaluate Environment Readiness
		Business Requirements Engineering (from Requirements Engineering process method fragment in OPF)

Table 19 continued

SDM	Task	Corresponding task method fragment(s)
	2. Service-Oriented Design	Identify Services
		Service-Oriented Architecture Engineering
	3. Service Development	Implement and Test Necessary Services
		Implement Necessary Wrappers
	4. Service Testing	Implement and Test Necessary Services
		Implement Necessary Wrappers
	5. Service Deployment	Publish Web Services
	6. Service Administration	Monitor Operational Web Services
		Compose Web Services Dynamically
	NA = 6; SMF = 16; MC = 6/16 (0.375)	
IBM RUP for SOA	1. Service Identification	Identify Services
	2. Service Specification	Specify Detail of Services
	3. Service Realization	Candidate Component Evaluation (OPF)
		Candidate Component Solution Identification in OPF (from Component Product Acquisition process method fragment in OPF repository)
	NA = 3; SMF = 16; MC = 3/16 (0.187)	
SUN SOA RQ	1. Inception	Evaluate Environment Readiness
		Business Requirements Engineering (from Requirements Engineering process method fragment in OPF)
	2. Elaboration	Evaluate Environment Readiness
		Service-Oriented Architecture Engineering
		Business Requirements Engineering (from Requirements Engineering process method fragment in OPF)
	3. Construct	Implement and Test Necessary Services
		Implement Necessary Wrappers
	4. Transition	Publish Web Services
	5. Maintenance	Monitor Operational Web Services
		Compose Web Services Dynamically
	NA = 5; SMF = 16; MC = 5/16 (0.312)	
SOAF	1. Information Elicitation	Business Requirements Engineering (from Requirements Engineering process method fragment in OPF)
	2. Service Identification	Identify Services
	3. Service Definition	Specify Detail of Services
	4. Service Realization	Candidate Component Evaluation
		Candidate Component Solution Identification in OPF (from Component Product Acquisition process method fragment in OPF repository)
	5. Road Map and Planning	Develop Governance Model for Current Iteration
		Plan Transition
	NA = 5; SMF = 16; MC = 5/16 (0.312)	
Steve Jones' Service Architectures	1. Initiate	Plan Transition
		All tasks in Project management process method fragments in OPF
		Business Requirements Engineering (from Requirements Engineering process method fragment in OPF)

Table 19 continued

SDM	Task	Corresponding task method fragment(s)
	2. Create Big Picture	Evaluate Environment Readiness
	3. Create Architecture	Service-Oriented Architecture Engineering
	NA = 3; SMF = 16; MC = 3/16 (0.187)	
Papazoglou	1. Planning	Plan Transition
		All tasks in Project management Process method fragments in OPF
	2. Analysis and Design	Evaluate Environment Readiness
		Identify Services
		Specify Detail of Services
	3. Construction and Testing	Implement and Test Necessary Services
		Implement Necessary Wrappers
	4. Provisioning	Develop Necessary Composite Web Services
		Discover Necessary Web Services
	5. Deployment	Publish Web Services
	6. Execution and Monitoring	Monitor Operational Web Services
		Compose Web Services Dynamically
	NA = 6; SMF = 16; MC = 6/16 (0.375)	
SDM proposed by Chang and Kim	1. Identifying business processes	Evaluate Environment Readiness
		Business Requirements Engineering (from Requirements Engineering process method fragment in OPF)
	2. Defining Unit services	Identify Services
		Specify Detail of Services
	3. Discovering Services	Discover Necessary Web Services
	4. Developing Services	Publish Web Services
	5. Composing Services	Develop Necessary Composite Web Services
	NA = 5; SMF = 16; MC = 5/16 (0.312)	

during SDM construction. This is not feasible given the time constraint of our research and the unavailability of real projects. Consequently, we expect that the real validity of our proposed fragments should be appraised in the long term. However, our earlier research in this area [67] suggested strongly that original service-oriented SDMs that have been utilized for identifying method fragments have already attested the suitability and applicability of tasks or, better stated, method fragments because they had been derived from recurrent pre-examined best practices. Therefore, we can assume that our proposed method fragments have been validated too implicitly.

However, to provide a more concrete measurement and explicit evidence on the usability of the proposed method fragments, we conducted two case studies. By usability in the context of SME, we mean how much do method fragments satisfy the requirements of an SDM [98]. We define a simple intuitive metric wherein the satisfaction of the requirements of an SDM is defined as the percentage of the number of requirements that are met by method fragments divided by the number of all requirements as shown in Eq. 3.

$$\text{Usability}(\%) = \frac{M}{R} \times 100 \tag{3}$$

In Eq. 3, M represents the number of requirements met, R represents the total number of SDM's requirements, and *Usability* represents the percentage of the usability of method fragments. High *Usability* means that method fragments have met most of the SDM's requirements. A 100% *Usability* means that method fragments have met all of SDM's requirements. We can thus measure the usability of the proposed method fragments in each case study (i.e., real project) using this criterion.

The two case studies presented here demonstrate how the proposed method fragments were used in the construction of a specific service-oriented SDM based on the enhanced OPF repository. In both case studies, a method engineer first elicited SDM requirements and then designed an SDM by selecting relevant method fragments from the repository. Both case studies were focused on the process of selecting method fragments rather than performing all steps involved in the construction of an SDM.

8.3 First case study

8.3.1 Scenario

The first case study is the development of a service-oriented system for providing some residential services to employees of an NGO [87]. The NGO has offices in 30 provinces with a total number of 14,000 employees. Based on the business process viewpoint, the system should provide online services for booking rooms and accepting payments for the expenses. After deploying the system, any employee can send his/her request to book a room in one of the hotels located in a specific province and track his/her request and pay the expenses by online services provided by third party payment services. Having received the requests, the priorities are automatically determined by the system and a room is assigned to the employee. The employee is informed by the system through e-mail and SMS services that confirm the reservation process.

8.3.2 SDM requirements

The aim is to satisfy the SDM requirements via appropriate method fragments in order to design the required methodology. Efforts aiming at developing any SDM should begin by clearly defining what the situational requirements of such SDMs are. Method engineers are responsible for mapping the elicited high-level requirements of the project to method fragments. For simplicity, we envisaged a direct mapping between SDM requirements and method fragments [21]. When the SDM requirements were fixed, method engineers clarified SDM requirements as shown in Table 20. Stakeholders have imposed some requirements. For instance, business processes modeling and improvement were forced due to the explicit request of stakeholders to receive a detailed documentation of their as-is and to-be business processes. Other requirements were relevant to SDM quality such as agility of development process, fast responsiveness to business volatility, flexibility, time, and cost of system development.

8.3.3 Method fragment selection

To illustrate how the proposed method fragments can be really incorporated during construction a service-oriented SDM, we confined ourselves to a simple manual process for method fragments selection, rather an automatic method fragments selection with an ontology flavor [17]. To realize such SDM requirements through method fragments, method engineers started by setting the overall development life cycle at the highest level of abstraction by using the *Business Optimization Phase, Initiation Phase, Construction Phase, Delivery Phase, Usage Phase, Retirement Phase* method fragments. All other fine-grain method fragments were placed into phase method fragments. After that, method engineers elaborated the SDM using task method fragments. To do this, method engineers took a set of consecutive inference and decisions based on the requirements and their relations with task method fragments. By considering the sections of each task method fragments, method engineers figured out which task(s) matched a requirement. Table 21 synopsizes how each requirement has been satisfied through one or more method fragments. According to this table, analyzing each requirement signifies one or more work products that should be produced to realize a target requirement. So, a method engineer selects relevant task method fragments to achieve the required work products. It should be noted that all method

Table 20 SDM requirements

Identifier	Name	Explanation
#R1	Utilizing External Services	Organization decided to use third party e-bank services to supply chain of business processes
#R2	Improving Business Process	The improvement of residency business processes was imperative
#R3	Using Legacy Systems Services	In order to reduce cost and effort of system development, potential legacy functionalities should be reused. In this regard, a number of old Fox Pro resident systems existed irrespective of being out of date
#R4	Modernizing Legacy Systems	Existing NGO legacy system and related operational databases should be modernized without stopping the current business processes. Traditional databases should be replaced by novel technologies
#R5	Conforming to Stated Quality of Services	Quality of external Web Services, specifically full availability and rate of discount per transaction are essential requirements
#R6	Provide Residency as Service	The residency business process should be exposed as a service to external consumers
#R7	Requirements-Based	Elicited requirements should be considered in the development of services and consequently the target system. A past unsuccessful experience in NGO domain has shown that a miss-understanding of requirements has lead to the development of a useless system

Table 21 Selected tasks versus SDM requirements

Requirement			Mapping requirements to relative method fragments	
Identifier	Analyzing the Requirement	Deduced Required Work Products	Relative Task Method Fragment(s)	Supportive Technique
#R1	Utilizing external services need to look after for the most appropriate Web-Services. Next, a contract with external supplier to remain on acceptable of QoS should be contracted. Web Services should be monitored during the usage to prevent degrading of QoS	A list of candidate Web Service should be discovered on the web For selected Web Services a consensus between service provider and consumer should be contracted Web Service should be observed during the usage	Specify SLA Discover Necessary Web Services Monitor Operational Web Services	Create SLA contract Search Web Services Monitor the QoS of Web Services
#R2	The current business processes should be modeled, analyzed and re-engineered wherever an improvement is urgent	Modeling current business models Make improvement on the business process	Process Needs Assessment Process Tailoring Process Mandating	Existing Techniques [14,15] Existing Techniques [14,14] Existing Techniques [13,15]
#R3	The feasibility and practicality of currently deployed legacy systems should be assessed whether business logic of existing logic can be wrapped with Web Service technologies while data reside on them	A list of candidate business logics can be wrapped into Web Services technology State of readiness NGO's infrastructure	Evaluate Environment Readiness	Create a Readiness Report
#R4	While the new system has significant impact on through of the NGO so modernization strategies and alternative solutions should be assessed	Producing an approved strategy or more strategies to migrate to a new service-oriented system	Plan Transition	Make Transition Plan
#R5	External Web Service that called via NGO system should be monitored continuously. Ones that work improperly and violate from theirs contracts should be replaced with new Web Services	Need to monitor procedure for Web Services adopted in system according to contracts	Compose Web Service dynamically Specify SLA	Reconfigure Composite Web Services Create SLA Contract
#R6	The goal of the requirement is to decompose booking and paying business process into set of service in or to achieve integrity and reusability of process	A list of candidate service that from residency business process	Identify Services Specify Details of Services	Top-Down Bottom-Up Add Specific Details to Services
#R7	Software's requirements should be formally elicited, documented into requirement engineering documents and then validated by all stakeholders	A list of identified and prioritized software requirements and requirements models	Requirements Identification Requirements Prototyping Requirements Specification Stakeholder Profiling Technology Analysis	Existing Techniques [14,15] Existing Techniques [15,15] Existing Techniques [14,14] Existing Techniques [13,15] Existing Techniques [14,15]

fragments need not be included in a project-specific SDM due to project requirements.

The existing OPF repository can be used for requirement elicitation, specification, and validation. For such tasks, some of the existing general techniques have been adopted, which are most commonly used in any situation and so are incorporated in the constructed SDM. Selection of other tasks is based on the SDM requirements. For instance, method engineers select the *Specify Service Level Agreement (SLA), Discover Necessary Web Services, and Monitor Operational Web Services* tasks to satisfy #R1. For improving existing business processes, OPF contains numerous tasks that help business processes to be partially or fully optimized. These tasks that are placed in the Business Optimization Phase

method fragment assist method engineers to explore organization business processes and re-engineer them based on needs (refer to #R2).

While a number of residency systems have been developed independently in the organization and have now become obsolete, the *Evaluate Environment Readiness* task is selected to assess the documents of the legacy systems to see if they have any asset that can be reused (refer to #R3). The task had significant effect on reducing cost and time of development. Moreover, the old residency system's databases contained a large amount of history records about employees that should have been made available to the new system without losing their integrity. In this regard, the *Plan Transition* task was selected (refer to #R4). As the last functional requirement that the custom SDM should be supported, the *Compose Web Service Dynamically* was selected to satisfy #R5. For instance, e-bank services were replaced by other services, while the availability of the current service provider was reduced. Selection of some method fragments was unavoidable due to the special situation of the project. For instance, the selection of the *Identify Services* and *Specify Details of the Services* tasks were due to defining and exposing residency business processes as services (refer to #R6).

Having determined the overall development process via selection of appropriate method fragments, we had to show how the chosen tasks had to be performed Method engineers concretized each selected task by associating it with a specific supportive technique (Table 20). For example, to define the right services, method engineers associated *Top-down* and *Bottom-up* approaches to the *Identify Services* task.

For brevity, responsible roles and related artifacts are not shown in Table 21; they should be defined in real situations. The important point to note is that the resulting methodology must be further refined and adapted iteratively by method engineers during the maintenance of the system, in accordance with the project situation through iterative process reviews of the development process.

We can now empirically validate the usability of the proposed method fragments using Eq. 3. According column 1 of Table 20, the number of SDM requirements was 7. In addition, as shown in column 1 of Table 21, the number of requirements satisfied by one or more method fragments was also 7. According to Eq. 3, the percentage of requirements satisfaction is 7/7, meaning that all the requirements had been met by the proposed method fragments (100% usability).

8.4 Second case study

8.4.1 Scenario

We have chosen the Driver Assistance System (DAS) presented in [100] as our second case study. In contrast to the first case study that we did really implement in the context of a real software development project, we did not implement the second case study in real and just used it to show conceptually whether its SDM requirements were satisfied by our proposed method fragments or not.

DAS is categorized in the domain of real-time automotive systems that have high potential for SOSD utilization. DAS considers a target system with a number of sensors assisting the driver to monitor the safety features of the car such as the engine oil level, pressure of the cylinder heads, and the locking status of the doors. Sensors are equipped with safety critical embedded programs that check the status of that car and report potential failures or mishaps to the driver by triggering the execution of workflows composed of Web-Services that orchestrate Web-Services to aid the driver to decide what to do. For example, DAS aids the driver to select a suitable car service such as a garage, a tow truck, or a rental service in the area based on the received data from the car GPS system before or upon failures or crashes. The driver may specify preferences such as the desired garages, acceptable road and traffic conditions, affordable repair costs, and possible methods of money payments. On the other hand, DAS may know about some services such as accessible car service companies, truck companies, and parts retailers. Guided by such information provided by DAS, the driver can order appropriate services while diagnostic data about the car status is sent automatically to service providers, e.g., to dispatch spare parts to the driver location. Figure 5 shows an abstract schema of DAS. It is assumed that a safety critical real-time subsystem in the core of DAS checks the status of the car engine periodically and keeps an updated list of available car services.

8.4.2 SDM requirements

We have assumed that the company developing hypothetical software for DAS has set a new policy to migrate from traditional development of software from scratch (i.e., design, implementation, and test) to an assembling approach by using existing services to reduce the time and effort required to

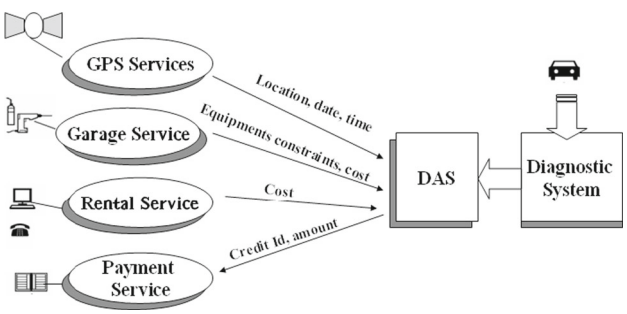

Fig. 5 An abstract schema of DAS

develop the software. Some situational factors have led the company to set this new policy. We have also assumed that most of the software developers in the company are expert and experienced in the development of data-intensive information systems rather than real-time systems. In addition, there are budget restrictions.

From the SME point of view, the development team must define a situational SDM in which a set of consecutive tasks aid them in the development of DAS. The method engineer should designate a situational SDM that meets the requirements of stakeholders in a timely and reasonable manner. The method engineer is also responsible for defining the SDM requirements and map them to relevant method fragments. Table 22 lists the key SDM requirements that the method engineer has identified.

8.4.3 Method fragment selection

We again confine ourselves to a simple manual process for the selection of method fragments, rather an automatic selection process. In line with situational factors of the company, the method engineer creates a composed service by using of a set of available fine-grain Web-Services. The main effort of the method engineer is thus spent on finding a set of relevant services to be intertwined together.

To develop a new SDM, the method engineer decides on the lifecycle of the SDM by selecting from phase method fragments, namely the *Initiation Phase, Construction Phase, Delivery Phase*, and *Usage Phase* method fragments. He/she then completes the details of the SDM by using the proposed task method fragments. To do this, the method engineer analyzes the sections of each task method fragment and figures out which task(s) match a requirement. It should be noted that #R1, #R2 and #R3 requirements are similar to each other, allowing the method engineer to select the same task method fragment for them all. Table 23 synopsizes how each requirement is satisfied by one or more proposed method fragments.

Because the proposed set of method fragments mainly focus on SOSD aspects, the method engineer cannot find any relevant support for #R5 and #R6. Therefore, #R5 and #R6 requirements remain unsupported by the method fragments and must be developed by the company from scratch. In fact, this is an example describing why the OPF repository should be enhanced with specific method fragments for real-time development.

According to column 1 of Table 22, the numbers of SDM requirements are 6, while the numbers of requirements met by the method fragments are 4 (column 1 of Table 23). According to Eq. 3, the percentage of requirements satisfaction is therefore 4/6 implying 66% usability.

The two case studies presented here demonstrated how the proposed method fragments were used in the construction of a specific service-oriented SDM based on the enhanced OPF repository. In both case studies, a method engineer first elicited SDM requirements and then designed an SDM by selecting relevant method fragments from the repository. Both case studies were focused on the process of selecting method fragments rather than performing all steps involved in the construction of an SDM.

9 Conclusion and future works

In this research work, we presented a set of new service-oriented method fragments that were derived from prominent service-oriented SDMs. These method fragments conform to the OPEN metamodel. We showed how method engineers could select appropriate fragments from the enhanced repository of OPF to construct project-specific service-oriented SDMs effectively.

In this work, we used a number of supportive techniques to derive our proposed method fragments. However, there is a need for more alternative techniques based on the project situation. Moreover, search for new method fragments as an ongoing process is needed. For instance, project man-

Table 22 SDM requirements

Identifier	Name	Explanation
#R1	Constraint on budget	No code must be implemented from scratch, except for trivial parts. Stakeholders have mandated to utilize as much as independent and available reusable Web-Services to construct the software through assembly and reduce the cost of development
#R2	Deploying minimum number of developers	As a sub requirement derived from #R1, stakeholders have decided to use a minimum number of developers
#R3	User preferences	Driver's preferences should be taken into account when selecting a car service
#R4	Risk of developer skill	The development team has little developers familiar with the Web-Service technology. They should thus use as many ready Web Services as possible
#R5	Implement and assemble hardware	The low-level code for sensors, timers, analog/digital converter, hardware wrapper, and I/O drivers should be designed, implemented, and tested. In addition, hosted hardware capability should be tested
#R6	Hardware Validation	Symbolic execution of hardware program should be performed to ensure correctness of code

Table 23 Selected tasks versus SDM requirements

Requirement			Mapping requirements to relative method fragments	
Identifier	Analyzing the Requirement	Deduced Required Work Products	Relative Task Method Fragment(s)	Supportive Techniques
#R1	In spite of development embedded code for sensors, other elements of the software systems should be provided via external service. So, obtaining Web-Services from outside reduce cost of project. This leads to searching Web-Services and composing them in order to satisfy user requirements. Developed system is a composite Web-Service that orchestrates a number of fine-grain Web-Services	A list of candidate Web Services in the Web should be discovered based on driver preferences SLA of the candidate Web-Service evaluated and those will be selected that satisfy driver requirements	Discover Necessary Web Services Specify SLA	Search Web Services Create SLA contract
#R2	This requirement has overlapping with #R1: Utilizing existing exposed Web-Service has significant impact on time and effort of software system development	A list of candidate Web Services that meet driver	Discover Necessary Web Services	Search Web Services
#R3	This requirement is similar to #R1 and #R2	Similar to R2	Similar to R2	Similar to R2
#R4	This requirement is similar to #R1 and #R2	Similar to R2	Similar to R2	Similar to R2
#R5	Not supported	–	–	–
#R6	Not supported	–	–	–

agement practices in SOSD need a new approach. Obviously, there are other service-oriented method fragments that we did not consider in our work. While a service-oriented software undergoes development by a number of possibly distributed development teams, it may raise new project management issues in terms of team management, cost, and effort estimation. Future work can thus enrich the proposed method fragments with more supportive service-oriented techniques and search for other necessary method fragments that are important in new situations and in new software paradigms. Applications of the proposed method fragments in the construction of service-oriented SDMs in real projects can also help refining and evolving the fragments, as well as validating it more fully. We did present two case studies to indicate how the proposed fragments can be used in the construction of SDMs, but more case studies especially in real projects are in order. A systematic assessment of the method fragments can be conducted through a hypothesize test that is a well-known technique for evaluating a proposed argument [100]. In short, this test evaluates how much the method fragments can be applicable and useful to software development organizations that use the proposed method fragments and those that do not use them during the construction of service-oriented SDMs. Of course, applying such a holistic test is very expensive, time-consuming, and thus was considered out of the scope of our research reported in this paper.

Acknowledgments We dearly thank Professor Brian Henderson-Sellers for his valuable comments on our research work. Only the third author's contribution to the research reported in this paper has been supported, in part, by the Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie)

References

1. Di Nitto, E., Ghezzi, C., Metzger, A., Papazoglou, M., Pohl, K.: A journey to highly dynamic, self-adaptive service-based applications. Autom. Softw. Eng. **15**(3–4), 313–341 (2008)
2. Papazoglou, P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: state of the art and research challenges. IEEE Comput. **40**(11), 38–45 (2007)
3. Tsai, W.T.: Service-oriented system engineering: a new paradigm. In: Proceedings of the IEEE International Workshop on Service-Oriented System Engineering (SOSE), pp. 3–6, Beijing, China, 2005
4. Qing. G., Lago, P.: Guiding the selection of service-oriented software engineering methodologies. Service Oriented Comput. Appl. 1–21 (2011). doi:10.1007/s11761-011-0080-0

5. Kumar, K., Welke, R.J. : Methodology engineering: a proposal for situation-specific methodology construction. In: Cotterman, W.W., Senn, J.A. (eds.) Challenges and Strategies for Research in Systems Development, pp. 257–269. Wiley, Chichester (1992)
6. Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Inf. Softw. Technol. **38**(4), 275–280 (1996)
7. Ralyte, J.: Towards situational methods for information systems development: engineering reusable method chunks. In: Proceedings of the 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education, pp. 271–282, Vilnius, Lithuania, 2004
8. Ralyté, J., Rolland, C. : An assembly process model for method engineering. In: Dittrich, K.L., Geppert, A., Norrie, M.C. (eds.) Advanced Information Systems Engineering. LNCS, vol. 2068, pp. 267–283. Springer, Berlin (2001)
9. Firesmith, D.G., Henderson-Sellers, B.: The OPEN Process Framework. An Introduction. Pearson Education Limited, Harlow (2002)
10. Nguyen, V.P., Henderson-Sellers, B.: Towards automated support for method engineering with the OPEN approach. In: Proceedings of the 7th IASTED SEA Conference, pp. 691–696. ACTA Press, Anaheim (2003)
11. Harmsen, A.F., Brinkkemper, S., Oei, H.: Situational method engineering for information systems projects. In: Olle, T.W., Verrijn-Stuart, A.A. (eds.) Methods and Associated Tools for the Information Systems Life Cycle. Proceedings of the IFIP WG8.1 Working Conference Cris/94, North Holland, Amsterdam, pp. 169–194 (1994)
12. Ralyté, J., Rolland, C.: An approach for method engineering. In: Proceedings of the 20th International Conference on Conceptual Modeling (ER2001). LNCS, vol. 2224, pp. 471–484. Springer, Berlin (2001)
13. Henderson-Sellers, B., Gonzalez-Perez, C., Ralyté, J.: Comparison of method chunks and method fragments for situational method engineering. In: Proceedings of the 19th Australian Software Engineering Conference (ASWEC2008), pp. 479–488, Los Alamitos, CA, USA, 2008
14. Firesmith, D.G., Henderson-Sellers, B.: The OPEN Process Framework. Addison-Wesley, London (2002)
15. Graham, I., Henderson-Sellers, B., Younessi, H.: The OPEN Process Specification, pp. 314–465. Addison-Wesley, London (1997)
16. Fahmideh, M., Jamshidi, P., Shams, F.: A procedure for extracting software development process patterns. In: Proceedings of the 5th UKSim European Symposium on Computer Modeling and Simulation (EMS), pp. 75–83 (2010)
17. Nguyen, V.P., Henderson-Sellers, B.: OPENPC: a tool to automate aspects of method engineering. In: Proceedings of the 16th International Conference on Software and Systems Engineering and Their Applications, ICSSEA 2003, Paris, France (2003)
18. Kumar, K., Welke, R.J. : Method engineering: a proposal for situation-specific methodology construction. In: Cotterman, W.W., Senn, J.A. (eds.) Systems Analysis and Design: A Research Agenda, pp. 257–268. Wiley, Chichester (1992)
19. Henderson-Sellers, B., Ralyté, J.: Situational method engineering: state-of-the-art review. J. Univ. Comput. Sci. **16**(3), 424–478 (2010)
20. Hofstede, A.H.M., Verhoef, T.F.: On the feasibility of situational method engineering. Inf. Syst. J. **22**(6/7), 401–422 (1997)
21. Saeki, M.: Toward automated method engineering: supporting method assembly in CAME. In: Workshop on Engineering Methods to Support Information Systems Evolution EMSISE'03, Geneva, Switzerland, 2003
22. Serour, M.K., Henderson-Sellers, B.: The Role of Organization Culture on the Adoption and Diffusion of Software Engineering Process: An Empirical Study Pearson/IFIP, pp. 76–88, Sydney, Australia, 2002
23. Henderson-Sellers, B., Edwards, J.M.: BOOKTWO of Object-Oriented Knowledge: The Working Object, pp. 594–634. Prentice-Hall, Sydney (1994)
24. Graham, I.: Migrating to Object Technology. Addison-Wesley, Wokingham (1995)
25. ISO/IEC Software Engineering: Metamodel for Development Methodologies, ISO/IEC 24744, International Organization for Standardization/International Electrotechnical Commission, Geneva, Switzerland, 2007
26. Informational Website on the OPEN Process Framework (OPF). http://www.opfro.org/
27. Henderson-Sellers, B., Hutchison, J.: (2003) Usage-centered design (UCD) and the OPEN process framework (OPF). In: Constantine, L.L. (ed.) Performance by Design. Proceedings of USE2003, Second International Conference on Usage-Centered Design, pp. 171–196. Ampersand Press, Rowley (2003)
28. OMG: OMG Unified Modeling Language Specification, Version 1.4, OMG Documents Formal/01-09-68 through 80 (13 Documents), 2001. http://www.omg.org
29. Firesmith, D., Henderson-Sellers, B., Graham, I.: OPEN Modeling Language (OML): Reference Manual. pp. 276–285. SIGS Books, New York (1997)
30. Nguyen, V.P., Henderson-Sellers, B.: Towards automated support for method engineering with the OPEN approach. In: Proceedings of the 7th IASTED Sea Conference, pp. 691–696. Acta Press, Anaheim (2003)
31. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley, London (2003)
32. Henderson-Sellers, B., Simons, A.J.H., Younessi, H.: The OPEN Toolbox of Techniques. Pearson Education Limited, UK (1998)
33. Henderson-Sellers, B.: An OPEN process for component-based development. In: Heineman, G.T., Councill, W. (eds.) Component-Based Software Engineering: Putting the Pieces Together, pp. 321–340. Addison-Wesley, Reading (2001)
34. Haire, B., Henderson-Sellers, B., Lowe, D.: Supporting web development in the OPEN process: additional tasks. In: Proceedings of 25th Annual International Computer Software and Applications Conference. COMPSAC 2001, pp. 383–389. IEEE Computer Society Press, Los Alamitos, CA, USA (2001)
35. Henderson-Sellers, B., Haire, B., Lowe, D.: Using OPEN's Deontic matrices for e-business. In: Rolland, C., Brinkkemper, S., Saeki, M. (eds.) Engineering Information Systems in the Internet Context, pp. 9–30. Kluwer, Boston (2002)
36. Henderson-Sellers, B., France, R.B., Georg, G., Reddy, R.: A method engineering approach to developing aspect-oriented modeling processes based on the open process framework. Inf. Softw. Technol. **49**(7), 761–773 (2007)
37. Debenham, J., Henderson-Sellers, B.: Designing agent-based process systems—extending the OPEN process framework. In: Plekhanova, V. (ed.) Chapter VIII in Intelligent Agent Software Engineering, pp. 160–190. Idea Group Publishing, Hershey (2003)
38. Henderson-Sellers, B., Debenham, J.: Towards OPEN methodological support for agent-oriented systems development. In: Proceedings of the First International Conference on Agent-Based Technologies and Systems, pp. 14–24. University of Canada, Canada (2003)
39. Low, G., Mouratidis, H., Henderson-Sellers, B.: Using a situational method engineering approach to identify reusable method fragments from the secure TROPOS methodology. J. Object Technol. **9**(4), 93–125 (2010)
40. BrescianiP. Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: TROPOS: an agent oriented software development methodology. J. Autono. Agents Multi Agent Syst. **8**(3), 203–236 (2004)

41. Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: High variability design for software agents: extending Tropos. ACM Trans. Autonom. Adapt. Syst. **2**(4), 16–27 (2007)
42. Henderson-Sellers, B., Serour, M.: Creating a Process for Transitioning to Object Technology. In: Proceedings of the 7th Asia-Pacific Software Engineering Conference, APSEC 2000, pp. 436–440. IEEE Computer Society Press, Los Alamitos, CA, USA, 2004
43. Serour, M., Henderson-Sellers, B., Hughes, J., Winder, D., Chow, L. : Organizational transition to object technology: theory and practice. In: Bellahse'ne, Z., Patel, D., Rolland, C. (eds.) Object-Oriented Information Systems. Lecture Notes in Computer Science (LNCS), vol. 2425, pp. 229–241. Springer, Berlin (2002)
44. Henderson-Sellers, B., Hutchison, J.: Usage-Centered Design (UCD) and the OPEN Process Framework (OPF). In: Constantine, L.L. (ed.) Performance by Design. Proceedings of USE2003, Second International Conference on Usage-Centered Design, pp. 171–196. Ampersand Press, Rowley (2003)
45. Fahmideh, M., Shams, M., Jamshidi, P: Toward a methodological knowledge for service-oriented development based on open meta-model. In: 2nd International Conference on Software Engineering and Computer Systems (ICSECS2011), Pahang, Malaysia, 2011
46. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: state of the art and research challenges. IEEE Comput. **40**(11), 38–45 (2007)
47. Lane, S., Richardson, I.: Process models for service based applications: a systematic literature review. Inf. Softw. Technol. (2010). doi:10.1016/j.infsof.2010.12.005
48. Ramollari, E., Dranidis, D., Simons, A.J.H.: A Survey of service-oriented development methodologies. In: The 2nd European Young Researchers Workshop on Service-Oriented Computing, Leicester, UK, June 2007
49. Fahmideh, M., Habibi, J., Shams, F., Khoshnevis, S.: Criteria-based evaluation framework for service-oriented methodologies, UKSim. In: Proceedings of the 12th International Conference on Computer Modeling and Simulation, pp. 122–130. Emmanuel College, Cambridge University, UK (2010)
50. Graham, S.: Building Web Services with Java, 2nd edn. SAMS Publishing (2005)
51. Papazoglou, M.P., Heuvel, W.V.D.: Service-oriented design and development methodology. Int. J. Web Eng. Technol. **2**(4), 412–442 (2006)
52. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., Holley, K.: SOMA: a method for developing service-oriented solutions. IBM Syst. J. **47**(3), 377–396 (2008)
53. SUN Microsystems: SOA RQ Methodology—A Pragmatic Approach. http://www.sun.com/products/soa/soamethodology.pdf
54. Kruchten, P.: Rational Unified Process: An Introduction, 3rd edn. Addison-Wesley, Reading (2003)
55. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change, 2nd edn. Addison-Wesley, Reading (2004)
56. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile Software Development Methods: Review and Analysis. VTT Publications, Finland (2002)
57. Allen, P.: The Service-Oriented Process. CBDI J. (2007). http://www.cbdiforum.com/reportsummary.php3?page=/secure/interact/2007-02/serviceorientedprocess.php&area=silver
58. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River (2005)
59. Keith, M.: SOMA, RUP and RMC: The Right Combination for Service-Oriented Architecture. IBM® WebSphere® User Group, Bedfont (2008)
60. Zimmermann, O., Krogdahl, P., Gee, C.: Elements of Service-Oriented Analysis and Design. IBM Corporation (2004). http://www-128.ibm.com/developerworks/library/wssoad1/
61. Mittal. K.: Service-Oriented Unified Process (SOUP) (2006). http://www.kunalmittal.com/html/soup.shtml
62. Chang, S., Kim, S.: A systematic approach to service-oriented analysis and design. In: Proceedings of the International Conference on Product-Focused Software Process Improvement (PROFESS 2007), pp. 374–388. Springer, Berlin (2007)
63. Jones, S.: A Methodology for Service Architectures. Capgemini UK Plc, UK (2005). http://www.oasisopen.org/committees/download.php/methodologyforServiceArchitectures-ASIS-Contribution.pdf
64. Erradi, A.: SOAF: An architectural framework for service definition and realization. In: Proceedings of the IEEE International Conference on Services Computing, pp. 151–158, Chicago, USA, September 2006
65. Ralyté, J.: Towards situational methods for information systems development: engineering reusable method chunks. In: Proceedings of ISD 2004, pp. 271–282, Vilnius, Lithuania, 2004
66. Ambler, S.W.: Process Patterns Building Large-Scale Systems using Object Technology. Cambridge University Press, Cambridge (1998)
67. Fahmideh, M., Sharifi, M., Jamshidi, P., Shams, F., Haghighi, H.: Process Patterns for Service-Oriented Software Development. In: Proceedings of the 5th IEEE International Conference on Research Challenges in Information Science (RCIS'2011), Guadeloupe, French West Indies, France, 2011
68. Meier, F.: Service-Oriented Architecture Maturity Models: A Guide to SOA Adoption, Meier Fabian, Student Thesis (2006)
69. SOA Maturity Model: Compass on the SOA Journey. http://www.soainstitue.org/articles/article/soa-maturity-model-compass-on-the-soa-journey.html
70. Almonaies, A., Cordy, J.R., Dean, T.R.: Legacy System Evolution Towards Service-Oriented Architecture. In: Proceedings of SOAME 2010, International Workshop on SOA Migration and Evolution, pp. 53–62, Madrid, Spain, March 2010
71. McBride, G.: The Role of SOA Quality Management in SOA Service Lifecycle Management (2007). http://ftp.software.ibm.com/software/rational/web/articles/soa_quality.pdf
72. Johnston, S.: UML 2.0 Profile for Software Services, IBM, Software Group. http://www.ibm.com/developerworks/rational/library/05/419_soa/
73. Perepletchikov, M., Ryan, C., Frampton, K., Tari, Z.: Coupling metrics for predicting maintainability in service-oriented designs. In: Proceedings of the 18th Australian Conference on Software Engineering (ASWEC'07), pp. 329–340, Melbourne, Australia, (2007)
74. Perepletchikov, M., Ryan, C., Frampton, K.: Cohesion Metrics for Predicting Maintainability of Service-Oriented Software, pp. 328–335. QSIC, Portland (2007)
75. Garlan, D.: Software Architecture. In: Finkekstein A. (ed.) A Roadmap in the Future of Software Engineering. ACM Press, NY (2000)
76. Canfora, G., Fasolino, A.R., Frattolillo, G., Tramontana, P.: Migrating Interactive Legacy Systems to Web Services. In: CSMR, pp. 24–36 (2006)
77. Canfora, G., Fasolino, A.R., Frattolillo, G., Tramontana, P.: A wrapping approach for migrating legacy system interactive functionalities to service oriented architectures. J. Syst. Softw. **81**(4), 463–480 (2008)
78. Stroulia, E., El-Ramly, M., Sorenson, P.G., Penner, R.: Legacy Systems Migration in CelLEST. In: ICSE Posters (2000)

79. Stroulia, E., El-Ramly, M., Sorenson, P.G.: From Legacy to Web through Interaction Modeling. In: ICSM, pp. 320–329. Montreal, QC, Canada (2002)
80. Sneed, H.M.: Wrapping Legacy Software for Reuse in SOA, Technical Report (2005). http://www.techrepublic.com/whitepapers/wrapping-legacy-software-for-reuse-in-a-soa/286064
81. Business Process Modeling Initiative: Business Process Modeling Language. http://www.bpmi.org
82. Leymann, F.: Web Service Flow Language (2001). http://www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
83. Andrews, T., Curbera, F., Dholakia, H., Goland, Y.: Business Process Execution Language for Web Services, Version 1.0.31. (2002). http://www.ibm.Comrdeveloperworksllibrary/ws-bpel
84. Rao, J., Su., X.: A survey of automated web service composition methods. In: Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004, San Diego, CA, USA, 6 July 2004
85. Dustdar, S., Schreiner, W.: A survey on web services composition. Int. J. Web Grid Serv. **1**, 1–30 (2005)
86. Henderson-Sellers, B., Qumer, A.: Using method engineering to make a traditional environment agile. Cutter IT J. **20**(5), 61–74 (2007)
87. IK Relief Foundation. http://emdad.ir/upload/crm
88. Henderson-Sellers, B., Gonzalez-Perez, C.: Towards the use of granularity theory for determining the size of atomic method fragments for use in situational method engineering. In: IFIP WG8.1 Working Conference on Method Engineering—ME'11, Paris, France, 2011 (in press)
89. Boehm, B.: Software Engineering Economic, p. 37. Prentice-Hall, Englewood Cliffs (1981)
90. Pressman, R.: Software Engineering. A practitioner's approach, 6th edn. pp. 388 McGraw-Hill, NY (2005)
91. Mili, H., Mili, F., Mili, A.: Reusing software: issues and research directions. IEEE Trans. Softw. Eng. **21**(6), 528–562 (1995)
92. Maiden, N.A., Sutcliffe, A.G.: Exploiting reusable specifications through analogy. Commun. ACM **35**(4), 55–64 (1992)
93. Purao, S., Storey, V.: APSARA: a web-based tool to automate system design via intelligent pattern retrieval and synthesis. In: Proceedings of the 7th Workshop on Information Technologies & Systems, pp. 180–189, Atlanta, GA, 1997
94. Han, T., Purao, S., Storey, V.C.: Generating large-scale repositories of reusable artifacts for conceptual design of information systems. Decis. Support Syst. **45**(4), 665–680 (2008)
95. Ramsin, R., Paige, R.F.: Process-centered review of object-oriented software development methodologies. ACM Comput. Surv. **40**(1), 1–89 (2008)
96. Kenzi, A., El Asri, B., Nassar, M., Kriouile, A.: A model driven framework for multiview service oriented system development. In: International Conference on Computer Systems and Applications, pp. 404–411. IEEE Computer Society, USA (2009)
97. Object Management Group, MDA Guide Version 1.0.1. OMG (2000)
98. Ralyté, J., Brinkkamper, S., Henderson-Sellers, B. (eds.): Situational Method Engineering: Fundamentals and Experiences. In: Proceedings of the IFIP WG 8.1 Working Conference, Geneva, Switzerland, September 12–14. IFIP International Federation for Information Processing, vol. 244. Springer, Boston (2007)
99. ter Beek, M.H., Gnesi, S., Koch, N., Mazzanti, F.: Formal verification of an automotive scenario in service-oriented computing. In: Proceedings of the 30th International Conference on Software Engineering (ICSE'08), Leipzig, Germany, pp. 613–622. ACM Press, NY (2008)
100. Perry, D.E., Porter, A.A., Votta, L.G.: Empirical studies of software engineering: a roadmap. In: Finkelstein, A. (ed.) Proceedings of the Conference on Future of Software Engineering. ACM Press, NY (2000)

Author Biographies

Mahdi Fahmideh Gholami received his M.Sc. degree in software engineering in 2007 from the Iran University of Science and Technology (IUST), Tehran, Iran. He received his bachelor's degree in 2004 in software engineering from Shahid Beheshti University, Tehran, Iran. His areas of interests include method engineering, software engineering, and distributed systems.

Mohsen Sharifi is an associate professor of software engineering currently chairing the Computer Engineering Department of Iran University of Science and Technology. He directs a distributed system software research group and laboratory. His main interest is in the engineering and development of distributed systems, especially in scientific applications. The development of a true distributed operating system is on top of his wish list. He received his B.Sc., M.Sc., and Ph.D. in computer science from the Victoria University of Manchester in the UK in 1982, 1986, and 1990, respectively. His home page is located in http://webpages.iust.ac.ir/msharifi/.

Pooyan Jamshidi received his M.Sc. degree in software engineering in 2007 from the Amir Kabir University (AKU), Tehran, Iran. He received his bachelor's degree in 2004 in software engineering from Amirkabir University, Tehran, Iran. His areas of interests include method engineering, software engineering, and distributed systems. He is currently studying software engineering as a Ph.D. researcher in 3Lero, The Irish Software Engineering Research Centre, School of Computing, Dublin City University, Dublin, Ireland.