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
© Springer-Verlag 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

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 (SDMs) 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 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 sup-

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.