
Yousef Baghdadi, Department of Computer Science, Sultan Qaboos University, Muscat, Oman

ABSTRACT

Method engineering techniques albeit proven effective for paradigms such as function, object and component, are pertinent to limited aspects of service orientation (SO). The comparison frameworks show that the produced methods neither conform to SO design principles nor to SOA, which is an issue. This paper proposes a framework to guide engineering methods for service-oriented software engineering. It describes a method by its aggregates and the relationships between elements such as Service Science (SS), SO, SOA, SOC, and Web. The paper also describes the guidance for method engineering. The framework consists of two layered categories of entities: (C1) conceptual foundation entities: SS, SO, and SOA, and (C2) realization infrastructure entities: SOC and Web. These entities request and provide services from/to each other. The framework comprehensively describes the SOSE environment, enforces the construct service with fundamental properties and principles, produces SOSE methods that address the challenges of software engineering, and benchmarks the existing SOSE methods.

Keywords: Frameworks, Method Engineering, Methods, Service Oriented Software Engineering

1. INTRODUCTION

Software system development methods such as structural, object-oriented, or component-based adopt computing constructs of subroutine, object, or component respectively as main building blocks. These constructs have been examined in different computing paradigms such as structured programming (Clemens et al., 1985; Dijkstra, 1968; Parnas, 1972), object-orientation (Booch, 1986; Gamma et al., 1995), and component-based (Clemens, 1995; Szyperski, 1997). Each of these computing paradigms has made a breakthrough in improving the field of software development by stressing out reuse, through separation of concerns and information hiding.

Service-Oriented (SO) constitutes an evolution of these paradigms; and a response to the growing needs for greater reuse, integration, and composition of IT services to support business flexibility and agility (Baghdadi, 2015).
Service Science (SS) aspires to shape the concept service with fundamental, generic properties (Spohrer et al., 2008), whereas, SO shapes the construct service with specific properties, from a computational perspective, and provides a set of design principles (Erl, 2008). These properties and principles enforce loose coupling and interoperability of services in addition to the principles of separation of concerns and information hiding, in order to allow services to be the basic components of service-oriented solutions, built with respect to an architectural style that is Service-Oriented Architecture (SOA).

Yet, building these solutions requires a new engineering approach; referred to as Service-Oriented Software Engineering (SOSE). It mainly concerns with developing methods that encompass (i) the engineering of the services as basic components, (ii) the engineering of compositions of services as composites, (iii) the management of services and compositions (Papazoglou et al., 2011), (iv) the quality of both services and compositions (Bianco et al., 2007), and (v) the engineering of business services for, at least, a defined SOA maturity (Welke et al., 2011).

SOSE has entailed, as expected, provision of new methods from both academia and industry. Most of the well-known methods, such as SOAD (Zimmermann et al., 2004), SDLC (Papazoglou and van den Heuvel, 2006), Sensoria (Wirsing et al., 2008), SOAF (Erradi et al., 2006), SOMA (Arsanjani et al., 2008), Erl’s methodology (Erl, 2009), claim their compliance with SOSE, namely a delivery strategy from different alternatives in order to comply with SO and SOA. Yet, comparison frameworks such as those developed by Baghdadi (2012a), Gu and Lago (2011), and Kohlborn et al. (2009) show that the output of these methods, when applied, does not fully comply with SO and SOA. Indeed:

- They do not consider SOA maturity models, which may guide the type of services and the service sourcing (Welke et al., 2011).
- They do not explicit each role in SOA. Their respective process concentrates on the provider of the service, while the consumers and the registry not only have a role but also constrain SOA (Baghdadi, 2013).
- They vary significantly in terms of their coverage to SOSE aspects. Some methods cover only services as components, whereas others cover components and composites. Very few of them attempt to cover business services (Arsanjani et al., 2008; Kohlborn et al., 2009).
- They vary in their delivery strategies such as meet-in-the-middle, top-down, bottom-up, or green-field (Papazoglou and van der Heuvel, 2006).

Indeed, the existing methods have been engineered by using one of the following approaches:

- Consolidation and integration of concepts, rules and practices, which has required a summarization of methods with respect to best practices, and comparison frameworks (Gu and Lago, 2011; Kohlborn et al., 2009)
- Extension of existing methods, from different paradigms, that have proven robust, such as UML in Sensoria project (Wirsing et al., 2008) or SOMA (Arsanjani et al., 2008)
- Evolution, by using a model or initial meta-model as-is as a basis for development to achieve the desired model to-be ((Ralyté et al., 2004).
- Product line and assembly techniques, i.e., reuse of fragments of methods (Brinkkemper, 1996, Brinkkimper et al., 1999; Mazo et al., 2012).

Some of these approaches, referred to as method engineering and situational method engineering, focus on the in-house construction of an organization specific or project specific methodological approach (Henderson-Sellers and Ralyté, 2010). They engineer methods by using existing, stored pieces of methods, namely fragments or chunks.
Mining Users' Intents from Logs
[www.igi-global.com/article/mining-users-intents-from-logs/126306?camid=4v1a](www.igi-global.com/article/mining-users-intents-from-logs/126306?camid=4v1a)

Multithreaded Programming of Reconfigurable Embedded Systems
[www.igi-global.com/chapter/multithreaded-programming-reconfigurable-embedded-systems/50424?camid=4v1a](www.igi-global.com/chapter/multithreaded-programming-reconfigurable-embedded-systems/50424?camid=4v1a)