Chapter 9
Addressing Highly Dynamic Changes in Service-Oriented Systems:
Towards Agile Evolution and Adaptation

Andreas Metzger
Paluno (The Ruhr Institute for Software Technology), University of Duisburg-Essen, Germany

Elisabetta Di Nitto
Politecnico di Milano, Italy

ABSTRACT

This chapter sets out to introduce relevant foundations concerning evolution and adaptation of service-oriented systems. It starts by sketching the historical development of software systems from monolithic and mostly static applications to highly-dynamic, service-oriented systems. Then, it provides an overview and more thorough explanation of the various kinds of changes that may need to be faced by service-oriented systems. To understand how such changes could be addressed, the chapter introduces a reference service life-cycle model which distinguishes between evolution, viz. the manual modification of the specification and implementation of the system during design-time, and (self-)adaptation, viz. the autonomous modification of a service-oriented system during operation. Based on the discussion of the key activities prescribed by that life-cycle, the chapter elaborates on the need for agility in both adaptation and evolution of service-oriented systems.

INTRODUCTION

For future software systems and software development processes, the only constant will be change. The "world" in which those future software systems operate is reaching unprecedented levels of dynamicity (de Lemos et al., 2011). Those systems will need to operate correctly in spite of changes in, for example, user requirements, legal regulations, and market opportunities. They will have to operate despite a constantly changing context that includes, for instance, usage settings, locality, end-user devices, network connectivity and computing resources (such as offered by Cloud computing).

DOI: 10.4018/978-1-4666-4301-7.ch009

Furthermore, expectations by end-users concerning the personalization and customization of those systems will become increasingly relevant for market success (Adomavicius & Tuzhilin, 2005).

Modern software technology has enabled us to build software systems with a high degree of flexibility. The most important development in this direction is the concept of service and the Service-oriented Architecture (SOA) paradigm (Erl, 2004; Kaye, 2003; Josuttis, 2007). A service-oriented system is built by "composing" software services (and is thus also called "service composition" or "composed service" in the literature).

Software services achieve the aforementioned high degree of flexibility by separating ownership, maintenance and operation from the use of the software. Service users do not need to acquire, deploy and run software, because they can access its functionality from remote through service interfaces. Ownership, maintenance and operation of the software remains with the service provider (Di Nitto, et al., 2008).

While service-orientation offers huge benefits in terms of flexibility, service-oriented systems face yet another level of change and dynamism. Services might disappear or change without the user of the service having control over such a change.

Agility, i.e., the ability to quickly and effectively respond to changes, will thus play an ever increasing role for future software systems to live in the highly dynamic "world" as sketched above. Agility can be considered from two viewpoints:

- First, agility may concern the evolution of the system. This means that it concerns the development process and how engineering activities (such as requirements engineering and implementation) should be performed to timely address changes by evolving the software.
- Secondly, agility may concern the adaptation of the system. This means that it concerns the system itself and how the system should respond to changes (Papazoglou et al., 2007). Agility in adaptation is typically achieved through self-adaptation, i.e., the autonomous modification of a service-oriented system during operation.

In this chapter, we first sketch the historical development of software systems from monolithic and mostly static applications to highly-dynamic, service-oriented systems. Then, we provide an overview and more thorough explanation of the various kinds of changes that need to be faced and how these could be addressed. As reference for the remainder of the chapter, we then introduce a service life-cycle model which integrates evolution and adaptation into a coherent framework. After elaborating on the activities prescribed by that life-cycle, we discuss the need for agility in evolution and adaptation. We conclude this chapter with our perspectives on agile development for service-oriented systems.

HISTORICAL DEVELOPMENT

The Emergence of the SOA Paradigm

In (Di Nitto et al., 2008) we gave an extensive account of the historical development of software technology and methods toward highly dynamic, service-oriented systems. The following paragraphs briefly summarize the major milestones along this development.

Genesis: In the late 1960s software development processes started to get disciplined through the identification of well-defined stages and criteria, which were to be met in order to progress from one stage of the process to the next. The waterfall life-cycle model as proposed by Royce in 1970 was such an attempt. It was very rigid and advocated the need for software developers to focus not only on coding but also on higher-level