Chapter 4
Service-Oriented Computing Applications (SOCA) Development Methodologies: A Review of Agility-Rigor Balance

Laura C. Rodriguez-Martinez
Tecnológico Nacional de México/IT Aguascalientes, Mexico

Hector A. Duran-Limon
CUCEA, Universidad de Guadalajara, Jalisco, Mexico

Manuel Mora
https://orcid.org/0000-0003-1631-5931
Autonomous University of Aguascalientes, Mexico

ABSTRACT

Software development methodologies (SDMs) have had an accepted evolution (i.e., the replacement of SDMs of one era to the next) through the pre-methodology and early-methodology eras to the methodology era. But in the last 20 years, the transition of the methodology era (rigor-oriented) to the post-methodology era (agile-oriented) has led a debate on benefits and drawbacks of rigor vs. agile orientation. Regarding the general software-engineering evolution, the service-oriented software engineering (SOSE) that studies service-oriented computing (SOC) development approaches, which are widely used to develop software-oriented computing applications (SOCA), has emerged. SOSE developers then face the problem of selecting and adapting a SOCA SDM. This chapter compares 11 SOCA SDM on agility-rigor balance by a framework of Boehm and Turner addressing the rigor-agility conflicts by defining three factors and their methodological characteristics. Each characteristic is evaluated for each SDM with a novel agility-rigor 45-point scale. Results suggest three of such SDMs are agility-rigor balanced.

DOI: 10.4018/978-1-7998-4165-4.ch004

INTRODUCTION

Software development tasks have used methodologies that can be tracked from four eras (Avison & Fitzgerald, 2003; Rodriguez et al., 2008). The pre-methodology era where no software engineering methodology was available only supported a generic programming approach. In the early-methodology era the first software engineering methodologies such as Waterfall (Royce, 1970) and SADT (Dickover, McGowan & Ross, 1977) emerged. The methodology era included more comprehensive and rigorous software engineering methodologies such as Spiral (Boehm, 1988), RUP (Kruchten, 2004), and MBASE (Boehm et al., 2004) . Lastly, the post-methodology era involves the agility approach (Dyba & Dingsoyr, 2009), which emerged in the last 20 years with Scrum (Sutherland & Schwaber, 2011) and XP (Beck, 1999) as the main development methodologies.

According to Avison and Fitzgerald (2003) and Rodriguez et al. (2008), the transition and evolution from one era to the next one has been well-accepted from developers and project managers because an era advanced on new development technology paradigms (e.g. object-oriented programming languages) and/or advanced on new required project management knowledge for a better project control over the previous era. This seamless evolution occurred from the pre-methodology era toward the early-methodology one, as well as from this one toward the methodology era. However, the last transition from the methodology era (based on a rigorous approach) toward the post-methodology era (based on an agile approach) has had some methodological conflicts and debates (Boehm, 2002; de Marco & Boehm, 2002).

The methodology era prescribes the need of rigorous methodologies to control and document practically "everything" that occurred in the project. However, such rigor (also called heavy-process methodologies) approaches imply a heavy weight processes, which can be complex to follow. Such methodological difficulties fostered the emergence of a counter-approach based on agility. Nevertheless, this emergent agile development approach, while reducing drastically the "everything" is in control, has been identified as a hard practice to perform it correctly. According to Beck and Boehm the agile development approaches rely on a "technically premier team" (2003), and thus "agility is only possible through greater discipline on the part of everyone involved." (Beck & Boehm, 2003; pp. 44). Consequently, current software developers and software project managers face on one hand the difficulty to correctly execute an agile development process -regarding its proper characteristics, like implicit quality controls, and the need of high creativity programmers-, and on the other hand the difficulty to execute a rigorist -heavy or highly planed or controlled- development process that emphasizes the documentation and the control of tasks rather than the creativity on the development process and the evolvability of the developed product (Sommerville, 2005). Additionally, Beck & Boehm (2003) reported that a lack of rigor on development methodologies lead to failed software projects although agile development methodologies avoid the bureaucratic documentation issues. Given that both the rigorous and agile approaches have presented benefits and drawbacks, balanced methodologies for software development have been proposed (Beck & Boehm, 2003; Boehm & Turner, 2004). As Beck and Boehm (2003; pp. 46) report, there are required "efforts to synthesize the best from agile and plan-driven methods to address our future challenges of simultaneously achieving high software dependability, agility, and scalability."

This chapter addresses such a topic (i.e. the rigorous-agile balanced software development approach) but from a relatively new development technology perspective of Service-Oriented Software Engineering / Service-Oriented Computing SOSE/SOC (Papazoglou et al., 2006; Rodriguez-Martinez et al., 2012). SOSE research stream investigates systematic, disciplined and quantifiable approaches to develop service-