Requirements for Software Process Modeling Tools

Rodrigo M. Pagliares

Computer Science Department – Federal University of Minas Gerais (UFMG)
Av. Antônio Carlos, 6627 – CEP 31.270-010 – Belo Horizonte – MG - Brazil
pagliares@gmail.com

Abstract. To be done.....

1. Introduction

Software Development is all about process. In the last years, several software processes were developed and used [Boehm 1996], [Krutchen 2003], [Jacobson+ 1999], [Humphrey 1995], [Schwaber 2004], [Beck 2004]. A portion of these tried to impose some form of rigor in the way which software is conceived, documented, developed and tested [Pfleeger and Atlee 2005]. The other ones were conceived, in the late 1990s, by some developers who had resisted to this rigor and formulated their own principles, trying to highlight the roles that flexibility could play in producing software quickly and capably. They codified their thinking in “the agile manifesto” [Agile 2009] that focuses on alternative ways of thinking about software development.

Several process and software process definitions can be found on literature. Accordingly to IEEE, a process is a sequence of steps performed for a given purpose [IEEE 1990]. More specifically, Ian Sommerville defines a software process as a set of activities whose goal is the development or evolution of software [Sommerville 2006]. Roger S. Pressman states that software process is a series of predictable steps – a roadmap to be followed helping the creation of timely and high quality results [Pressman 2001].

A software process can be modeled in natural language, by graphical descriptions or using software process meta-models [Nardini+ 2008], [Combemale+ 2006]. Henderson-Sellers [Henderson+ 2004] depicts various benefits of software process modeling:

- ensure a consistent, reproducible approach to all projects providing an uniform proposal to software development with guidance;
- control and support the whole development life with good project planning and management;
- allow consistency and traceability though the whole life cycle;
- emphasize analysis and understanding through customer involvement;
- reduce risk associated with shortcuts and mistakes;
- produce complete and consistent documentation from one project to the next;
- gain flexibility to process tailoring and managing and supporting different types of projects.

Since the 80s, there have been various process meta-models and finally OMG (Object Management Group) decided to create a standard in this area. The specification of SPEM (Software & Systems Process Engineering Meta-Model), version 1.0, was 
published in 2002. In 2005, SPEM 1.1 was available with minor updates. SPEM 1.1 still had several shortcomings; for example, semantics were ambiguous and hard to understand and there was a lack of enactment support. SPEM 2.0 was released in 2007 fixing the flaws in previous versions, being compliant with UML 2 and providing guidance on migrating existing process models from SPEM 1.1 to SPEM 2.0 [OMG 2008].

SPEM is a standardized way of expressing any software development process. The specification was developed especially to address the unique and complex nature of software development. It is vendor, framework and methodology neutral and leverages the expressiveness and popularity of UML (Unified Modeling Language) [OMG 2009]. The SPEM 2.0 is a meta-model developed by OMG, described through a meta-modeling language (meta-meta-model) [OMG 2006], [OMG 2008].

With so many benefits and a standardized language for process modeling, ones would argue that the direct use of SPEM for modeling process would be a natural choice. However, as stated by Wilson de Pádua Paula Filho, the SPEM 2.0 metamodel is of huge complexity and hard to be used directly by process developers [Paula Filho 2009]. To ease this complexity, a process modeling tool becomes necessary. Until now, few of these were developed but, with the advent of SPEM 2.0, is my belief that a plethora of them will be available in a near future.

Several people would benefit from using or constructing these tools: process designers could more easily describe and improve the existing company’s software development process; software development team members would have, for instance, some level of consistency and shared language across the organization; software engineering instructors could use these tools to teach current best practices in their curriculums and also bring the state-of-the-art software best practices to the software industry.

As so many people would benefit from software process modeling tools, a catalog with requirements of such ones would be of great help, aiding team members, instructors, project managers and process designers to compare different implementations on the market and helping tools providers to direct future developments. Matthias Hoffmann, Nikolaus Kuh, Matthias Weber and Margot Bittner presented a catalog of requirements for requirements management tools in the area of automotive as well as aircraft and defense systems [Hoffmann+ 2004]. The objective of this work is to provide a similar catalog in structure but with focus on software modeling tools instead.

The remainder of this paper is organized as follows: in section 2, we present software modeling tools models and features. Section 3 describes the requirements for such tools. Practical and operational aspects are presented in Section 4. Section 5 lists some tools, already on market, and how well they adhere to the requirements presented at Section 3. Finally, in Section 5, conclusions are made.

References


Humphrey, W. S. (1999), Introduction to the Team Software Process. Addison-Wesley


