Chapter 2.5
Tool-Support for Software Development Processes

Marco Kuhrmann
Technische Universität München, Germany

Georg Kalus
Technische Universität München, Germany

Gerhard Chroust
Kepler University Linz, Austria

ABSTRACT

Software development projects are complex. The more complex a project is, the higher are the requirements related to the software development process. The implementation of a process is a great challenge. This, in part, has to do with human factors (acceptance, etc.) as the benefits of a formal development process might not be obvious immediately and it may take a while until the process becomes the lifeblood of a team. A crucial step towards implementing, enacting and enforcing a process is to provide tool support for the many activities the process asks for. Tool support is necessary to guarantee efficiency in the project, to do the housekeeping and to minimize the "overhead" of the process. This chapter describes challenges and options for supporting process models by tools. Furthermore it describes concrete samples and shows how tool chains can be created with commercial tools as well as with open source tools.

SOFTWARE ENGINEERING ENVIRONMENTS

The quality demands on to-day's software together with the demands on the development processes are increasing due to the growing complexity of today's software products. In answering these demands the guidance of software development by a software development process model and the support of all activities by tools has nowadays become state-of-the-art, leading to so-called *software engineering environments* (SEE). Due to increased awareness of project risks and cheaper software and hardware, the utilization of SEEs becomes interesting not only for large but also for small and medium enterprises (SME). Traditionally tools always supported software development for certain development tasks like code-generation,

DOI: 10.4018/978-1-61350-456-7.ch2.5

design support etc. The understanding that the development process needs also to be incorporated and supported has brought about tools to support and enforce the enactment of software processes. Especially for a SME the choice of the right process and appropriate tools is difficult. A highly dynamic project environment, agile development practices, strict time-to-market requirements etc. are common in SMEs. Nonetheless, SMEs might find themselves in situations that necessitate and justify the introduction of a more formal process instead of only being agile. Such situations could be:

- In the last years, capability maturity models (e.g. CMMI (Chrissis, Konrad, & Shrum, 2006)) have gained popularity. Along with that development, many customers nowadays demand their suppliers to use standardized, certified processes.
- A company or a team could grow to a size that makes it hard to apply a purely agile development methodology.
- Distributed development with teams being geographically dispersed asks for more explicit structure in the development process (Raghvinder, Bass, Mullick, Paulish, & Kazmeier, 2006).

In this chapter we discuss options, limits and challenges of process enactment. We discuss what to consider when implementing processes in organizational environments with respect to organization-structure, users and tools. We then list options of process/tool-integration. Topics of interest are capabilities of software development processes, options for bringing together processes and tools and finally the area of process-aware tools. A central question in this area is: "What is adequate tool-support for a development process?" Finally in this chapter we give two real world examples using common and widely spread collaboration and development tools. The examples concentrate on two points of view in a software development project: The first example targets the management as user audience and the second one aims at supporting developers. A more general discussion of options for integrating standard and open source tools completes this section.

Terminology and State of the Art

Tool support always existed for certain development tasks (editors, compiler, flow charters etc.). With the growth of the complexity of software products and the necessary development processes also other tools had been introduced. Up until quite recently, software development processes were mostly described verbally in the form of books or articles. The implementation of a verbally defined process had to rely heavily on educating project members. Little to no tool support was available and would have been hard to realize because no machine-understandable description of the process was available. Much of the enactment and enforcement of a process thus had to be done manually by the people in a project.

Two schools of thought can be roughly distinguished since several years:

- One proclaims the post-bureaucratic age and relies on agile methodologies such as XP (Beck & Andres, 2004; Cockburn, 2001) and Scrum (Schwaber & Beedle, 2008). These lightweight processes contain a couple of core concepts and leave much of the details of a project unspecified. The rationale being that a software development project is so full of uncertainties and subject to change that an a priori process will be hard to follow anyway.
- The other one has driven forward the evolution of heavier processes – up to a level where these processes are described in a formal model. Examples for this kind of development processes (or process frameworks) are the German V-Modell XT (Koordinierungs- und Beratungsstelle der