Goals and Requirements for Supporting Controlled Flexibility in Software Processes

Ricardo Martinho, Polytechnic Institute of Leiria, Portugal

Dulce Domingos, University of Lisboa, Portugal

João Varajão, University of Trás-os-Montes e Alto Douro, Portugal

ABSTRACT

Software processes are dynamic entities that are often changed and evolved by software development team members. Consequently, flexibility is one of the most important features within software processes and related tools. However, in the everyday practice, team members do not wish for total flexibility. They prefer to learn about and follow controlled flexibility advice, that is, previously defined information on which, where, how and by whom they can change software process representations to match real-world situations. In this paper, the authors define a set of goals and requirements for a language and supporting software tool to control the flexibility within software processes. They follow a two-step approach, where 1) process engineers use the language constructs and supporting tool to define controlled flexibility-related information within software process models, and 2) software team members browse and learn from this information, and perform changes accordingly.

Keywords: Controlled Flexibility, Goal, Language, Model, Process, Requirement, Software

INTRODUCTION

Software processes represent a specifically ordered and organised set of the elements and relationships which are involved in the development of software products. The main elements that compose such processes are *activities*, *agents*, *artifacts*, *roles* and production support *tools*.

DOI: 10.4018/irmj.2010070102

Descriptions of these processes can be captured and converted into process models. Process engineers usually recur to a Process Modelling Language (PML) to define and represent these models. They facilitate human understanding and communication, and provide guidance for software development team members when executing the process.

Throughout the last three decades, several PMLs and supporting Process-centred Software Engineering Environments were developed to elicit process models and automate their support (see Bandinelli et al., 1994; Wise, 2006).

However, and as opposed to many stable business processes (such as production line-based ones), software processes are commonly held as dynamic entities, which often must be modified and evolved to cope with changes occurred, for instance, in the requirements of a certain software product, in the software organisation's structure or in the rapid changing software market (Cugola, 1998). Software process *flexibility* refers, precisely, to the ability to change parts of a process, without completely replacing it (Soffer, 2005).

However, more recent research advocate that, in the everyday business practice, most people do not want to have much flexibility, but would like to follow very simple rules to complete their tasks, making as little decisions as possible (Bider, 2005, Borch & Stefansen, 2006). In fact, latest case studies on flexibility in software processes (Cass & Osterweil, 2005) make evidence on the need of having software process engineers expressing and controlling the amount of changes that the remaining team members can or cannot make in the software process.

This controlled flexibility can be defined as the ability to control the way changes are to be performed, taking into account:

- Which process elements, as for example choosing which activities, roles or work products can or cannot be changed;
- At which abstraction level(s) of modelling (*where*) can or cannot those elements be subjected to changes. For example, the process engineer can require that changes made to the model representation of a *Test Solution* activity should be immediately reflected to all the software project plans (*instance* level) and real-world projects (*real-world* level) where this activity is referred;
- What are the dimensions of change involved (*how*), including, for example, which *operations* can or cannot be performed (such as *add*, *delete*, *move* or *skip*) and which *properties* will the change enclose (such as its *duration* and *extent*);
- Who can or cannot enforce those changes, including, for example, single users or role-based permissions.

In this paper we present a set of goals and requirements regarding a modelling language and proper software tool support to enable the representation of controlled flexibility information in software processes. This requires an in-depth understanding of software development social organisations, their work, and the ways cooperation and learning are enforced. Therefore, each derived goal and corresponding requirement(s) is supported by a set of needs and assumptions identified by important works from empirical software and knowledge engineering research areas.

This paper is organised as follows: the next section presents the research process we adopted for the development of a controlled flexibility-aware language and supporting software tool. It comprises the goal and requirements' definition activities and related specifications. The third section contains these specifications, as well as thorough reviews and justifications for the viewpoints expressed by each goal and requirement. Then, we present most prominent related work, and finally we conclude the paper and present future work.

THE PROCESS OF ELICITING GOALS AND REQUIREMENTS

We adopted the research process in Figure 1 to conduct the main activities involved in the analysis and development of a controlled flexibility-aware language and proper software tool support. It is an iterative and incremental process, with seven main activities. These activities sometimes overlapped, and feedback flows occurred between them, to foresee adjustments and consistency checks between their resulting work products.

For instance, the definition of goals was interleaved with the specification of requirements, which in turn alternated forwards and backwards with language and software tool