Chapter 20

Software Process Models are Software Too: A Domain Class Model for Software Process Models

Daniel Turk
Colorado State University, USA

Vijay Vaishnavi
Georgia State University, USA

A software process model describes a set of partially-ordered sequences of activities that are carried out in order to accomplish certain goals. While numerous process modeling approaches have been proposed over the years, it seems that none of them have yet addressed the full range of concepts described in this definition. Most focus on activities and activity ordering; few, if any, focus on representing organizational goals and process improvement. Most provide concrete implementation approaches; few, if any, describe the general model(s) upon which these implementations are built. This paper suggests a new approach for developing software process modeling systems.

If “Software Processes are Software Too,” as suggested by Osterweil (1987), then so are software process models, and hence there may be processes and models that are used in regular software development that may be useful in the process modeling domain as well. The paper focuses on the domain class model as an example of one type of model that might be produced if an approach such as the Unified Process were used in the process modeling domain.
domain. Such a process, and the set of models produced by it, if used, may be helpful in moving the process modeling field forward. While identifying the conceptual needs of process modeling systems, these models leave totally open the choice of how to formalize and implement actual solutions. A domain class model for process models is developed as an example of one of these models.

INTRODUCTION

A software process model describes a set of partially-ordered sequences of activities that are carried out in order to accomplish certain goals (Curtis, Kellner, & Over, 1992; Feiler & Humphrey, 1993). While numerous process modeling approaches have been proposed over the years (Arbaoui & Oquendo, 1994; Barghouti & Kaiser, 1992; Conradi et al., 1992; Curtis, Kellner, & Over, 1992; Dowson, 1987, 1993; Engels & Groenewegen, 1994; Finkelstein, 1989; Humphrey, 1989; Humphrey, 1995; Jarke et al., 1998; Kellner, Briand, & Over, 1996; Lehman, 1997; Paulk et al.,1995; Roland et al., 1995; Starke, 1994; Sutton & Osterweil, 1997; Workflow Management Coalition, 1994), it seems that none of them have yet addressed the full range of concepts described in this definition. Most focus on activities and their ordering; few, if any, focus on representing organizational goals and process improvement. Most provide concrete implementation approaches; few, if any, describe the general models upon which these implementations are built. This paper suggests a new approach for developing software process modeling systems.

In his landmark paper, Leon Osterweil (1987) stated that “software processes are software too.” Thus was launched more than a decade of work whereby software processes have been modeled as computer programs. The focus has been on using computer languages, developing new ones or extending old ones, to model software processes in an executable manner. This dynamic approach is believed to allow better modeling of software development processes than more static approaches can provide.

However, it seems that an important implication of Osterweil’s assertion has gone largely unnoticed: If software processes are software too, and thus are modeled by computer programs, then so are software process models, and hence the processes used to develop regular software might be valuable to use in developing process modeling software as well.

The Unified Process (Booch, Rumbaugh, & Jacobson, 1999; Rumbaugh, Jacobson, & Booch, 1999; Jacobson, Booch, & Rumbaugh, 1999), and most other approaches, suggest that in software development a number of models are useful: domain, application, test, etc. The domain model describes in a system-
To Prevent Reverse-Engineering Tools by Shuffling the Stack Status with Hook Mechanism
www.igi-global.com/article/to-prevent-reverse-enginnering-tools-by-shuffling-the-stack-status-with-hook-mechanism/126613?camid=4v1a