Software Process Modeling Using Object-oriented Programming and Rule-based Reasoning

Benny P. Phillips, Ph.D. and Don D. Haley
Oklahoma City Air Logistics Center
OC-ALC/LASFB Tinker AFB OK 73145

Abstract - In the context of software process modeling (process definition & simulation), this paper presents an object-oriented programming and rule-based reasoning method to simulate a part of an existing software process model. Presently, our process simulation emphasizes process training and enactment. That is, the simulation is on the "what and how" portion of the model.

I. INTRODUCTION

In the time of increasing software complexity, development costs and maintenance effort, much attention has been devoted to software technologies for improving software processes [1]. New software process technologies focus on the areas of process assessment, modeling (definition and simulation), training and enactment.

This paper presents a computer method to simulate a part of an existing software process model, which we have found suitable for use in the Operational Flight Program group of the OC-ALC/LAS division (LAS/OFP). The model that we selected is given by Basili and Rombach [2]. It is known as the Tailoring A Measurement Environment (TAME) model.

By using object-oriented programming and rule-based reasoning (object-rule) method, Oivo and Basili [3] simulated a part of the TAME model before. They were interested in the Experience Base portion of the model. We now use a similar approach, however, instead of working with the Experience Base, we simulate activities in the "what and how" part of the TAME model (i.e., the planning phase) with emphasis on training and enacting the software development process, which is currently going on in the LAS/OFP.

Before discussing our partial implementation of the TAME model, in the following, we will briefly review essential ideas presented in [2], [3] to provide our readers some background in Software Engineering Process Modeling.

In June 1988, Basili and Rombach introduced TAME as a framework for software development and maintenance. For a given software project, the authors suggested four phases: characterizing, planning, executing and learning (the Experience Base). Each of the first three phases must be carried out constructively and analytically.

The characterizing phase of the model is required for understanding conditions that influence a software project. It is a required starting point for process and product improvement since we have to know "what" and "where" we are before talking about improvement. The characterization must be viewed analytically since an organization, for a given project, cannot only talk about the environmental conditions but examine them based on known facts from earlier projects.

The planning phase of the model is required for understanding what we want to do and how we are going to do it. In the "What" part, some of the constructive goals we can address are issues of on-time delivery, user satisfaction, etc. Analytically, we can design analysis procedures for supervising whether the goals are met. In the "How" part, constructively, things we can consider are execution methods, tools, etc. Analytically, execution methods and tools that we choose here should allow us to do the project in a way that we can inspect whether our goals are satisfied.

The executing phase of the model is where the software product is actually produced using methods and tools selected in the planning phase. Note that, in all phases, lessons learned must be well documented in the Experience Base for future use.

To support the proposed model, Basili and Rombach gave twenty-four principles that can be used to justify in detail every phase of TAME—Ten on Software Engineering and fourteen on Software Measurement. For example, to justify the need for the planning phase, one can use one of the ten Software Engineering principles (P2) to argue that "we need to formalize the planning of construction process in order to develop quality a priori. Without such plans the trial and error approach can hardly be avoided."

In late 1992, Oivo and Basili implemented the Experience Base, part of the TAME model, using the object-oriented programming and rule-based reasoning approach. They indicated that the object-rule approach had been used successfully in a number of recent software engineering projects. For a good review on the object-rule method of capturing knowledge and on simulation of sequential
networks (i.e., the “how” part of the model can be viewed as a sequential network), the readers are referred to Shlaer and Mellor [4] and Putzolu and Roth [5].

Note that commercially-available software tools, one of which is STATEMATE [6], can be used to capture and simulate various aspects of software engineering. However, we decided to use the object-rule method because, first, we want to customize our simulation program in such a way that we can address the process training issue. Secondly, we want to develop this simulation project internally so that our people can learn more about the technical aspects of software process modeling. Therefore, any future upgrade should be manageable. For technical help, we use tips in [4], [5], [7] to represent the knowledge and temporal logic.

II. IMPLEMENTATION

A. Requirements Overview

Considering the current needs of the LAS/OFP, we believe that:

1) The simulation program must be easy to understand and use on a personal computer.
2) The simulation program must facilitate process communication, training and improvement.
3) The simulation program must be developed internally so that our people can get more technical understanding about software process modeling; therefore, future modification and growth will be easy.

B. Implementation

To effectively facilitate process communication, training and improvement we provide two parts in the implementation: 1) a hyper-textual-graphical description of the software development and maintenance process, and 2) a simulation program.

1) The description of the software development and maintenance process will be available in the form of a manual being prepared by the LAS/OFP. We want to convert the electronic copy of this manual into a portable, hyper-textual-graphical document similar to the help file used under Windows™ 3.1. The procedure required for this conversion is quite simple: convert the electronic copy of the manual into a special format, then use a help compiler to produce the portable, hyper-textual-graphical help file.

2) As in [3], we use the object-rule method to represent the “how” information contained in the manual. Based on the rules captured, we then simulate software engineering activities against organizational goals (e.g., the “what” information). We accomplish this task by using techniques suggested in [4], [5] and an expert system tool called KAPPA-PC™ [7].

ACKNOWLEDGMENT

The authors express appreciation to David Corder of the OC-ALC/LAS division for supporting this project. For technical discussion and helpful suggestions, the authors are grateful to John Holcomb of the LASF branch, Don Rauch of the LASM branch, Tom Saunkeah of the LASFB section and Ronald Wallman, James Sieber and Kelly Butler of the LAS Software Engineering Process group.

REFERENCES