

An Application of a Combined Software Simulation of a Software Development Process

Robert H. Martin
Portland State University

Dr. David Raffo
Portland State University

In today's competitive software development environment, improving software quality and time to market are essential. As software companies become more mature, it is natural for them to create a consistent software development process. Evolving and improving this process is an important way to increase quality and reduce time to market.

Simulation models of a software development process offer one technology that can be used to evaluate possible changes to the software process. Detailed process models allow managers to describe hypothetical process changes, to develop quantitative estimates of the impact of process changes, and to develop financial analyses of the impact of these changes on the business.

Existing process models have been developed based upon paradigms dictated by available simulation tools. Models of the project environment have been based upon system dynamics tools such as Dynamo and Stella. These models represent the project environment as a set of differential equations. Integrating these equations over time describes the behavior of project variables such as staff levels, motivation, and the number of detected errors. While these models are excellent ways to demonstrate the effects of feedback loops which may exist in the project environment, they are inherently limited in their ability to represent discrete process steps.

Discrete event models are an appropriate tool to represent individual process steps. These models allow direct representation of the process tasks and are designed to facilitate the description of process entities through descriptive attributes. A discrete event software process model can include individual process steps for design, coding, and test. These models can also describe the code modules acted upon by the steps in terms of size, complexity, amount of reused code and so forth. Discrete event models do not easily allow the description of the project environment because time does not advance at constant intervals.

State based models are an excellent way to describe systems with concurrent activities but suffer the same problems as discrete event models in describing the project environment.

By modifying existing simulation software tools, we have created a software process model that allows both discrete events simulation to describe process steps and continuous simulation to represent the differential equations that describe the project environment. In this paper, we describe the application of this modeling capability to a large software development process.

In this application, we have developed a software model of an existing 71-step software development process. Over a two-year period, we collected data on the duration and effort for each process step. Using extensive interviews, we developed a quantitative description of the project environment.

In this paper, we demonstrate how this detailed model may be used to examine the impact of two potential changes.

Abdel-Hamid and Madnick's systems dynamics model of the project environment includes the sector that models changes in workforce. In this widely accepted model, staff members are hired based on a hiring rate that may vary with changes in the desired workforce level. New staff members are then assimilated into the project at a specified rate that is based on the average amount of time required to learn both the tools and the application. During this assimilation, some staff members may be transferred to other projects. In a large company conducting several simultaneous large projects, there is a potential to transfer experienced staff members into the project thus avoiding assimilation delays. We wanted to develop a quantitative estimate of the impact of this ability to transfer experienced staff into the project. Our conjecture was that this capability should decrease project duration and improve software quality.

A second area of investigation involved the value of the unit test process step. In this process code modules were initially tested by the developer during the unit test step. Modules were then integrated into larger units called "processes" which were formally tested by the quality assurance department in the "process test" step. Deleting the unit test step was expected to increase the time required for process test and to increase the number of errors detected and reworked. The purpose of this investigation was to determine whether the advantages of deleting the unit test step are more significant than the disadvantages of increased time and errors in the process test step.

Because our combined model integrates both continuous and discrete event simulation capabilities, we can examine the simultaneous effects of both changes to transfer policy and the deletion of the unit test step. In other words, we may examine the hypothesis that the ability to quickly increase the experienced staff level allows us to produce software with so few errors that the unit test process is unnecessary.

This application demonstrates that a combined model of the software development process is a realistic tool for examining questions that are beyond the capability of either continuous or discrete event models.