

Assessing the Impact of Various Defect Reduction Practices on Quality, Cost and Schedule

Ioana Rus

Fraunhofer Center for Experimental Software Engineering, Maryland
3115 Ag./Life Sciences Bldg. #296
University of Maryland, College Park, MD 20742
(301) 405-8340
irus@fc-md.umd.edu

James S. Collofello

Arizona State University, Computer Science and Engineering Department
collofello@asu.edu

The problem addressed in this paper is how to assess and predict the impact of different defect reduction software engineering practices on product quality and on project effort, cost, and schedule. We propose using a simulator of the software development process as an aid to deciding what software engineering practices to use for developing a product to meet the required quality attributes and also what would be the impact of implementing such practices on effort and schedule.

The model was designed based on hierarchical building blocks that correspond to the main phases of a development process, starting at a higher level and specializing as necessary. A modular, flexible, and extensible modeling package was thus obtained. This library of modules was used to model a waterfall life cycle project but other types of development processes can be also modeled, as needed.

The intended users of the simulator are software engineers and managers who have to plan and control the development of a software product with specified quality requirements, and have project time, resources, and budget constraints.

The purpose of the model is understanding, training, planning (policy assessment and decision making), and tracking. The scope and boundaries of the model are:

- one project (not multiple projects or company level)
- new product, entirely developed by the company
- addresses only development and does not include the maintenance phase.

The modeling technique used is system dynamics modeling. The tool support existing for this modeling approach allows the model to be executed, simulating a real project.

The model would help answer questions such as:

- How does the usage of different practices impact on project cost?;
- How does the usage of different practices impact on delivery time?;
- How much effort must be allocated to the project to deliver it with the required reliability value/range?;

- Is it cost effective to implement a specific software engineering practice?;
- What are the dynamics of defects during the development cycle?;
- How does the reliability of the software change if a specific practice is performed?;
- How much does the increase of quality cost?

The model was implemented using continuous modeling in *Extend* simulation environment.

This paper presents the design and implementation of the model. A software development process is considered to have a set of production phases (such as requirements analysis and specification, design, and coding) and a system testing phase. The main flows throughout the lifecycle correspond to artifacts and defects.

A generic production phase has the following activities: Production, Verification and Validation (V&V), and Rework. In every production phase items from previous phase are used to produce new artifacts which are verified and validated, and then sent out to the next phase. During production, defects are injected into products. Some of the defects injected in the current phase together with defects propagated from previous phases are detected during the V&V activity. The undetected defects will propagate to the subsequent phases. The detected defects are reworked and new defects (bad fixes) might be generated and propagate to the next phases.

During system testing phase test cases are executed, given execution time constraints. When a failure is encountered the corresponding faults are identified and then corrected. Regression testing is performed periodically to detect possible new defects introduced (bad fixes) that will also be reworked. The defects that do not manifest and are not detected will remain in the delivered product and will affect the field value of reliability. The simulator has a reliability model integrated, in order to capture failure rate evolution and relate it to the remaining defects.

Defect reduction software engineering practices are modeled through their influence on different factors that affect defects. For instance defect prevention practices contribute to reduction of defects injected in the product but they might also reduce the productivity. Therefore, these practices will take more time and effort in production, but the time and effort spent later for detecting and reworking these defects is eliminated. Defect detection practices increase the number of defects detected and do not let them propagate to subsequent phases, where the cost to fix them is much higher. Fault tolerance practices mask some of the defects and do not allow them to manifest when the software is executed, so the number of failures caused by the number of defects remaining in the final product is reduced.

An example of how to use the simulator is given by presenting a couple of *what-if* scenarios for predicting how varying defect detection and correction activities could impact on defects, effort and schedule. We present the results of simulating three hypothetical projects. The hypothesis we test here is that performing defect detection and correction earlier in the life cycle ultimately saves effort and time and improves the quality of the final product. The paper presents the simulation results that confirm the hypothesis and also some unexpected behavior of the simulator that leads to a better understanding of the dynamics of a software project. This also shows how a computer model can be used as a tool to support the “gut feeling” of a decision maker in software development projects.

Keywords

Software process modeling and simulation, software quality, defect reduction practices, project planning, defect model, system dynamics modeling.