Integrating Metrics into a Large-Scale Software Development Enviroment

Sallie Henry and John Lewis

TR 89-1

Integrating Metrics into a Large-Scale Software Development Environment

Sallie Henry and John Lewis
Computer Science Department, Virginia Polytechnic Institute and State University, Blacksburg, Virginia

Software metrics have been shown to be indicators of software complexity. These complexity indicators identify error-prone, unmaintainable software, which if identified early can be rewritten and/or thoroughly tested. However, the integration of metric use into commercial situations faces practical resistance from budget and deadline restrictions. This paper presents an experiment that introduces a nondisruptive method for integrating metrics into a large-scale commercial software development environment.

1. INTRODUCTION

A major goal of software producers is to develop reliable, maintainable code. However, given that software development is necessarily guided by budget and time constraints, the criteria for internal software acceptance is often that it satisfy the requirements of specific tests. Unfortunately, correct software according to the test plan does *not* guarantee good software in terms of reliability or other quality factors [1].

The key to integrating quality into software stems from the relationship between code complexity and errors. As the complexity level of a piece of software increases, the code becomes difficult to understand and is more likely to contain logical defects. Code containing errors must be modified, but that maintenance is nontrivial since the code is difficult to comprehend. To compound the problem further, it has been shown that programs cannot be made more maintainable by simply changing their code. In fact, Belady and Lehman established that maintenance activities tend to *increase* the level of complexity of the code, so more maintenance is likely to be required [2].

Therefore, the effort to instill quality characteristics into software must be a major concern during the development process. Many software engineering techniques have been proposed to decrease the complexity of software, and consequently increase reliability and other quality factors. Unfortunately, most of these techniques are difficult to measure and enforce. If maintainability is to be an expected consequence of software development, ways to measure it must become an integral part of the development cycle. Therefore, a subjective, quantitative evaluation of software complexity is needed.

Software metrics, when defined and used correctly, have been shown to be good indicators of software complexity [3]. Metrics are quantitative evaluations of software design or code based on some set of criteria that contribute to the software's complexity. If metric analysis results were required to conform to quality tolerances, the development process would not be compromised.

Software complexity metrics will not solve the entire problem of low-quality software. However, studies have shown that metrics can predict error proneness [4] and can indicate where a redesign effort is beneficial [5]. These aspects of software quality represent good points for concentrating specific improvement efforts.

This paper describes a methodology that uses software metrics to integrate some quality factors into large-scale software. The methodology is implemented in a commercial environment of a major software producer as a means to complement their software development process. This cooperation between commercial software developers and academicians is essential for validation of metric research results. This research complements past projects concerning metrics tool development [6], large-scale software metrics [7], and working relationships with other software development organizations [8].

One requirement of the project was the development of a tool and associated methodology that would operate unobtrusively to current software production, that automatically generated quantitative metrics, and that would provide threshold values for metric interpretation.

To test the validity of the model, the methodology is used to analyze source code frozen in a mid-development release point of a large project. The soft-

Address correspondence to Dr. Sallie Henry, Computer Science Department, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061.