Chapter 5
Enhancing Testing Technologies for Globalization of Software Engineering and Productivity

Amir H. Khan
University of Maryland, USA

Atif M. Memon
University of Maryland, USA

ABSTRACT

While successful at increasing code churn rates, global software development and evolution suffers from several quality assurance challenges. First, sub-groups within developer communities often work on loosely coupled parts of the application code. Each developer (sub-group) typically modifies a local "copy" of the code and frequently checks-in changes (and downloads other developers' changes). Consequently, after making a change, a developer may not immediately realize that the local change has inadvertently broken other parts of the overall software code. This situation is compounded as there is little direct inter-developer communication -- almost all communication is done via web-based tools such as code commit log messages, bug reports, change-requests, and comments. This chapter outlines the challenges that global software development adds to the already-complex quality assurance process. Two case studies of real software projects implemented in a disturbed manner demonstrate the importance of continuous integration testing and the positive consequences of increasing the diversity of quality assurance techniques/tools. Finally, it concludes with an outline of how software integration testing needs to be enhanced to meet the new challenges of globalization.

1. INTRODUCTION

As software becomes increasingly complex, pervasive, and important for the general consumer across the world, there is an unprecedented demand for new software. It is no longer practical to develop software in one part of the world and "export" it to other parts. Moreover, as new economies emerge and strengthen across the world with improved educational infrastructure, software development expertise is also becoming distributed. Together with improved global telecommunication networks, it is not surprising that the answer to today's software needs is sought in globalization of development.

DOI: 10.4018/978-1-60566-731-7.ch005

Globalization of software development has several advantages. First, it has led to unprecedented code churn rates; developers work in shifts around the world, leveraging the world's time zones to evolve commercial and open-source software around-the-clock. Second, it has enabled the development of large and complex software – software experts from around the world collaborate, sharing their experiences to solve difficult coding and design problems. Third, it has accelerated the improvement of software architectures – there has been an increased interest in component-based software applications – developers work on different loosely-coupled components simultaneously to build the software. Finally, it has led to new agile processes that advocate collaborative development.

Distributed and global development has also created new challenges, especially for quality assurance. Because sub-groups within developer communities work on loosely coupled parts of the application code, each developer (sub-group) typically modifies a local "copy" of the code and frequently checks-in changes (and downloads other developers' changes). Consequently, after making a change, a developer may not immediately realize that the local change has inadvertently broken other parts of the overall software code. This becomes worse because there is little direct inter-developer communication as almost all communication is done via web-based tools such as code commit log messages, bug reports, change-requests, and comments. Because of these challenges, a large number of multi-site software projects have suffered from significant quality issues (Porter, A., Yilmaz, C., Memon, A.M., Schmidt, D.C. & Natarajan, B., 2007).

The next section presents an overview of some techniques used to test evolving software. Sections 3 and 4 present case studies of two medium-sized multi-site software projects. These case studies describe the software testing related challenges encountered in those projects. They also help to further understand the various types of testing related challenges that future projects are likely to encounter as global software development becomes more complex. Finally, Section 5 outlines current directions in software testing research and practice that are helping to address the software testing challenges in a global development scenario.

2. TESTING TECHNIQUES FOR EVOLVING SOFTWARE

The ongoing quest of the software industry to keep software of all sizes integrated throughout the course of its development, has led to the mainstream adoption of many enabling practices. Nightly/daily builds and smoke tests (Karlsson, Andersson, & Leion, 2000), (McConnell, 1996a), (Olsson, 1999) have become widespread (Robbins, 2000), (Halloran, Scherlis, 2002). During nightly builds, a development version of the software is checked out from the source code repository tree, compiled, linked, and "smoke tested" ("smoke tests" are also called "sniff tests" or "build verification suites" (Marick, 1998)). Typically, unit tests (Robbins, 2000) and sometimes acceptance tests (Crispin, House, Wade, 2001) are executed during smoke testing. Such tests are run to (re)validate the basic functionality of the system (Marick, 1998). Smoke tests exercise the entire system; they do not have to be an exhaustive test suite but they should be capable of raising a "something is wrong here" alarm. A build that passes the smoke test is considered to be "a good build." As is the case with all testing techniques (Schach, 1996), it is quite possible that problems are found in a good build during more comprehensive testing later or after the software has been fielded.

Daily building and smoke testing have been used for a number of large-scale commercial and open-source projects. For example, Microsoft used daily builds extensively for the development of its Windows NT operating system (McConnell, 1996). By the time it was released, Windows