Open-Source Software Systems
Understanding Bug Prediction and Software Developer Roles

R. B. Lenin, University of Arkansas at Little Rock, USA

S. Ramaswamy, University of Arkansas at Little Rock, USA

Liguo Yu, Indiana University South Bend, USA

R. B. Govindan, University of Arkansas Medical Sciences, USA

ABSTRACT

Complex software systems and the huge amounts of data they produce are becoming an integral part of our organizations. We are also becoming increasingly dependent on high quality software products in our everyday lives. These systems 'evolve' as we identify and correct existing defects, provide new functionalities, or increase their nonfunctional qualities - such as security, maintainability, performance, etc. Simultaneously, more software development projects are distributed over multiple locations (often globally) and are often several millions of dollars in development costs. Consequently, as the Internet continually eliminates geographic boundaries, the concept of doing business within a single country has given way to companies focusing on competing in an international marketplace. The digitalization of work and the reorganization of work processes across many organizations have resulted in routine and/or commodity components being outsourced.

INTRODUCTION

Currently there is an increase in the total world-wide investment in research and a wider world-wide distribution of research and development activities. It is predicted that government action and economic factors will result in more global competition in both lower-end software skills and higher-end endeavors such as research. Critical drivers include lower costs and increase quality, member flexibility and unorthodox round-the-clock approaches to project execution, enhanced creativity in the development environment, the interdependency between economic and software development and much more. It has been argued that standardized jobs are more easily moved from developed to developing countries than are higher-skill jobs. Employees in this global environment need to comfortable with the theory, blend it with necessary practice by understanding the business and cultural issues involved; while being able to effectively share, communicate, articulate and advance their ideas for an innovative product &/or solution.

From a project management perspective, it is imperative that project managers be able to deal with such geographically separated diverse groups in an effective manner. This implies that they need to address two critical

DOI: 10.4018/978-1-60566-731-7.ch028

issues: *(i)* Resource planning / forecasting based upon the need for software maintenance, which is influenced by the number of *bugs* occurring in the various software components, and *(ii)* understanding the interaction patterns among the various software developers. In this chapter, we concentrate on these two specific issues, both relating to issues of staffing / resource allocation, that impact cost and influence effective project management, using data produced from open-source software (OSS) repositories:

1. **Prediction of future bugs:** Given the globalized nature of such software development projects and the associated costs; even a slight improvement in predicting the expected bugs can lead to immense improvements in effective project planning and management, and,
2. **Understanding the dynamics of distributed OSS developer groups:** Development personnel in groups who are also team builders, are critical to sustain a long-term geographically dispersed project development. Understanding the interaction dynamics between team members in a distributed software development team can be critical for issues relating to personnel and merit decisions.

1. CASE STUDY

1.1. Predicting Bugs in Open Source Software Systems

In this section, we discuss how software bugs can be predicted for open source software repositories, which are beginning to accumulate a large stash of usable data. This data can be effectively harnessed for significant productivity and efficiency gains in both software product development and software project management. Open source software bug repositories have been recognized as reliable data assets that can provide useful information to assist in software development and software project management. These repositories enable end-users to automatically submit bug information, as and when they are encountered. Moreover, these repositories help researchers mine the data to predict the occurrence of future bugs which helps in the optimization of available (limited) resources. Additionally in a competitive and rapidly changing world of software services delivery, it allows delivering quality products on time and within budget.

Open source software bug repositories have been recognized as reliable data assets that can provide useful information to assist in software development and software project management. Considerable attention has been paid in collecting such bug repositories in the recent past (Anvik, Hiew & Murphy, 2005; Bug reports for Eclipse projects, 2009; Mining Open Source Software Engineering Data 2008). These repositories enable end-users, who need not technically savvy in software development, to submit bug information, as and when they are encountered, in an automated manner. Moreover, these repositories can help researchers mine the data to predict future bugs which helps in the optimization of available (limited) resources. Additionally in a competitive and rapidly changing world of software services delivery, it allows delivering quality products on time and within budget (Hassan & Holt, 2005; Kagdi, Collard & Maletic, 2007; Lucr´edio, Prado & de Almeida, 2004).

Recently, a considerable amount of research has been done in **modeling** *bug* occurrence patterns. The data from open source repositories, including defect tracking systems are analyzed using probabilistic models to predict the files with *bug*s (Askari & Holt, 2006). A statistical model based on historical fault information and file characteristics was proposed to predict the files that contain the largest numbers of faults (Ostrand & Weyuker, 2002, 2004; Ostrand, Weyuker, and Bell 2004, 2005). Such information can be used to prioritize the testing components to make the testing process more efficient and the resulting software more dependable. In (Livshits & Zimmermann, 2005), the authors worked on the discovery of common method usage patterns that are likely