ABSTRACT

This paper presents an approach to prioritize program segments within the impact set computed using functional call graph to assist regression testing for test case prioritization. The presented technique will first categorize the type of impact propagation and then prioritize the impacted segments into higher and lower levels based on propagation categorization. This will help in saving maintenance cost and effort by allocating higher priority to those segments which are impacted more within the impacted set. Thus a software engineer can first run those test cases which cover segments with higher impacted priority to minimize regression test selection.

Keywords: Dynamic Analysis, Impact Analysis, Regression Test Selection, Software Maintenance, Software Testing

INTRODUCTION

The role of computer software has undergone significant change over a time span of little more than 50 years. Dramatic improvements in technology in hardware performance and computing architectures have led to more sophisticated and complex computer-based systems. One of the major challenges of software engineering is software evolution and its maintenance. Thus, there is an increased pressure on software engineers to produce high quality software in this revolutionary world. Studies indicate that 90% (Seacord, Plakosh & Lewis, 2003) of all software development is maintenance and more than 50% of the total maintenance cost of software lies in rework i.e. in changing the software (Lee, Offutt & Alexander, 2000; Myres, 1979). Making changes to the software without understanding and knowledge of the software component can produce disastrous effects (Bohner & Arnold, 1996) and can lead to degraded software.

Among the most challenging problems of software maintenance namely, program comprehension, impact analysis, and regression testing, software change management is one of the most essential tasks in software maintenance. Change impact analysis improves the accuracy of resource estimates, provides better scheduling, and can reduce the amount of corrective maintenance, because fewer errors

DOI: 10.4018/IJSSOE.2015040103
will be introduced. Whenever a change is made to any part of software system, it is important to gain complete understanding of structure, behaviour and functionality of the system being modified. The software maintenance process can only be optimized if precise and unambiguous information is available about the potential ripple effects of a change to an existing system. When a single part of software is changed, other parts of the software are possibly subject to change as well. To investigate which parts of the software are to be changed, change impact analysis is performed. Bohner (2002) identifies three basic software change activities: understanding software with respect to the change, implementing the change, and retesting the newly modified system. Changes to software can be introduced during the whole software engineering process. Software change impact analysis, often called simply impact analysis, is a family of approaches for addressing these problems (Bohner & Arnold, 1996; Arnold & Bohner, 1993; Loyall, Mathisen & Satterthwaite, 1997; Pfleeger, 1998; Turver & Munro, 1994). Impact Analysis is defined in the IEEE Standard for Software Maintenance as “impact analysis: Identifies all system and software products that a change request affects and develops an estimate of the resources needed to accomplish the change. This includes determining the scope of the changes to plan and implement work, accurately estimating the resources needed to perform the work, and analyzing the requested change’s cost and benefits” (ISO/IEC, 2006). Impact analysis has three primary benefits for a software maintainer: improved accuracy of resource estimates, and thus better maintenance scheduling and reduced change costs; a reduction in the amount of corrective maintenance because of fewer introduced errors and improved software quality (Moreton, 1990).

This paper presents an approach to prioritize program segments particularly functional segments within impact set computed using functional call graph to assist regression testing for test case prioritization. The proposed technique will be helpful in saving maintenance cost and effort by allocating higher priority to those segments which are impacted more than other segments within the impacted set. Thus a software engineer can first run those test cases which cover segments with higher impacted priority to minimize regression test selection. Most of the research in this area has focused mainly on computing impact sets whereas the proposed technique will first predict impact set and later prioritize the most impacted segments from others within the impact set. The main contribution of this work is as follows:

1. Computation of impacted segments using functional call graph
2. Categorization of type of impact propagation
3. Prioritization of impact segments into higher and lower levels based on propagation categorization
4. Minimization of test cases during regression test selection resulting in less maintenance cost.

These contributions are helpful in estimating the impact of such changes thereby aiding in selecting useful test cases for regression testing. The remainder of this paper is organized as follows: next section discusses related work followed by proposed approach, demonstration with the example to generate results followed by conclusion.

RELATED WORK

When planning modifications, impact analysis helps maintainers to predict effects and costs of planned changes. Broadly impact analysis techniques can be partitioned into two categories: (a) traceability analysis and (b) dependence analysis (Bohner, 2002). Traceability analysis refers to the ability to define, trace, and document relationships between software lifecycle objects such as requirements, design documents, source code, and other documentation (Cimitile, Fasolino & Visaggio, 1999). Impact analysis techniques based on dependence analysis attempt to assess the effects of change on semantic
Quality, Improvement and Measurements in High Risk Software
www.igi-global.com/chapter/quality-improvement-measurements-high-risk/77730?camid=4v1a

Multiformalism Modeling Compositionality in SIMTHESys
Enrico Barbierato (2014). Theory and Application of Multi-Formalism Modeling (pp. 81-103).
www.igi-global.com/chapter/multiformalism-modeling-compositionality-in-simthesys/91942?camid=4v1a