Chapter 64
An Exhaustive Requirement Analysis Approach to Estimate Risk Using Requirement Defect and Execution Flow Dependency for Software Development

Priyanka Chandani
Jaypee Institute of Information Technology, Noida, India

Chetna Gupta
Jaypee Institute of Information Technology, Noida, India

ABSTRACT

Requirement defects are one of the major sources of failure in any software development process as they prevent smooth operation and is taxing both in terms of tracking and validation. The objective of this article is to make requirement analysis phase exhaustive by estimating risk at requirement level using requirement defect information and execution flow dependency as early as possible to inhibit them from being incorporated in design and implementation. The proposed approach works as a two-fold process which computes risk involved with each requirement twice. The whole process is divided into a three-layered framework to finalize requirements with clear vision and scope of a project. The entire process has been supported by a software case study. The results of the proposed work are promising and will help software engineers in ensuring that all business requirements are captured correctly with clear vision and scope. It will also help in decreasing the chances of failure, risk, and conflicts between stakeholder and developer and other challenges involved to develop the project.

DOI: 10.4018/978-1-7998-3016-0.ch064

INTRODUCTION

Requirements engineering is the first and crucial phase of software development and all the subsequent phases are influenced by requirements (Denger & Olsson, 2005). It is a systematic approach to understand, formally describe, evaluate/validate and attain an agreement on the nature of the problem (Lamsweerde, 2000). Thus, making quality of requirements as one of the important factors for overall quality of the project (Alshazly, Elfatatry & Abougabal, 2014). To improve quality, requirements should be gathered, analysed, validated precisely and properly which really leads to determine the success of the project. Otherwise, the project ends up with poor requirements resulting in a crude design followed by wrong or unwanted focus in product development directly affecting the product and customers (Firesmith, 2007). Poorly specified, incorrect or missing requirements lead to defects in the system (Blackburn, Busser & Nauman, 2001). In fact, the most common types of defects in software development are requirement defects which are among the major sources of failure constituting 32.65% (Hamill & Katerina, 2009) and these defects have high severity problem which affect software maintainability (Chen & Huang, 2009). Durability of the project also comes into question when any software lifecycle stage cannot be backtracked to the root requirement fault. Hence, it becomes essential to work on requirement defects and its real causes at the early stage of software development.

In 1998, Card (Card, 1998) stated that if problems are classified or grouped, it becomes easy to identify the clusters in which errors are likely to be found. There are several classifications or taxonomies available for classifying requirements defects which help in creating more accurate and efficient defect prevention and detection techniques (Alshazly et al., 2014; Beizer, 1990; Chillarege et al., 1992; Grady, 1992; Margarido, Faria, Vidal & Vieira, 2011; Walia & Carver, 2009; Hayes, 2003). Historically, there has been lot of research on validating requirements using various techniques and some through defect taxonomies (Ackerman, Buchwald & Lewski,1989; Sommerville, 2004; Laitenberger, Atkinson, Schlich & Emam, 2000; Felderer & Beer, 2013, 2015). In practice, most defect taxonomies are used in the later stages of software development life cycle but the use of these taxonomies to validate requirements have not been fully exploited (Felderer & Beer, 2013, 2015) and only little has been done in the direction of linking and validating the requirements with defect taxonomy. Here, the focus is on relating the requirements which are categorized with the defect taxonomy and finding the risk associated with the core requirements.

To the best of our knowledge, there is no model where a risk is estimated for each requirement early in the software lifecycle based on defect information and requirements classification. The aim of this research is to estimate the risk associated with the implementation of each and every requirement based on (a) requirement defect information and (b) execution flow dependency as early as possible to inhibit them from being incorporated into design and implementation. The presented model will help software engineers to estimate the risk of implementing a particular requirement which as a result will help in ensuring that all business requirements are captured correctly with clear vision and scope and the solution is designed to meet its requirements by focusing on delivering value to the customer by taking a right decision using risk values towards its implementation. The presented approach will help in decreasing the chances of failure, risk, and chances of conflicts between stakeholder and developer and other challenges involved to develop the project. The proposed approach will have following benefits during requirement engineering phase of software development: