Optimal Release Policy for Multi-Release Software System

Anu G. Aggarwal, University of Delhi, Department of Operational Research, New Delhi, India
Chandra K. Jaggi, University of Delhi, Department of Operational Research, New Delhi, India
Nidhi Nijhawan, University of Delhi, Department of Operational Research, New Delhi, India

ABSTRACT

In software industry, multi release software development process is a latest phenomenon that brings the benefits of newer technologies, while retaining the quality. In this paper, it is assumed that the development of next version or release starts immediately after the launch of the previous version and the field test of each version continues after its release so that undetected faults of just previous version along with the added faults of latest version are detected during the testing of new software code. Today’s dynamic customers need timely up gradation. Therefore, to sustain user growth and satisfaction it is imperative for the developers to know the appropriate time to launch upgraded software into the market. In this paper, an optimal release policy for multi release software system has been proposed by taking into consideration the testing as well as the operational phase. A numerical example has been presented to illustrate the optimal release policy. Parameter estimation has been done using the real-life fault data set. Goodness-of-fit curves have been drawn.

KEYWORDS

Discrete SRGM, Operational Phase, Optimal release Policy, Software Reliability, Testing Phase

INTRODUCTION

The concept of software reliability modeling has gained lot of importance in recent years. Software reliability is a measuring technique for defects that causes software failures in which software behavior is different from the specified behavior in a defined environment with fixed time. Reliability is defined as the probability of failure-free software operation in a specified environment for a specified period of time. A software reliability growth model is known as one of the fundamental technologies for quantitative software reliability assessment. It is used to assess the reliability of the software during testing and operational phases. Software testing involves running the software and checking for unexpected behavior in software output. The successful test can be considered to be one, which reveals the presence of the latent faults. A software reliability growth model (SRGM) provides a mathematical relationship between time span of testing and the cumulative number of faults detected. Many SRGMs have been developed in the last two decades that can describe the reliability growth during the testing phase of the software development. In particular, non-homogeneous poison process (NHPP) based SRGMs are quite popular due to their mathematical tractability. Some of the popular SRGMs include models due to (Goel & Okumoto, 1979; Kapur, Pham, Gupta, & Jha, 2011; Pham, 2008; Robello & Goyal, 2010; Zhang & Mostashari, 2011). Many analytical SRGMs have been proposed reflecting testing and operational environment factors to predict the software reliability as it undergoes changes through the removal of failure causing faults. Based on some assumptions of the testing environment

DOI: 10.4018/IJORIS.2017070102

Copyright © 2017, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
one can estimate the number of faults that can be removed by the nth test case execution and hence the reliability. A large number of SRGMs assumed that operation profile can be treated as software testing profile. But in real, the user’s operational environment is not always same as that assumed in the software development phase (Musa, 1993). Some failures happen during testing phase which testing team removes immediately, while few others appear during operational phase of the previous version which testing team removes as reported from field. The interactions between the testing phase and operational phase for multi-release software is shown in Figure 1 (all figures are in the Appendix). In the operational phase, user may experience different failure occurrence rate over the time. Hence there is a need to recognize these two different concepts so that appropriate choice of models for fault removal phenomenon of each phase can be made.

The present software development environment is very competitive and advanced. Every customer needs a more efficient and error free software. As such software delivered with full functionalities and very high reliability built over a period of time may turn out to be unsuccessful due to technological obsolescence. Moreover, due to time and resource limitation, software firms do not attempt to deliver a complete and perfect product in one development cycle instead they offer successive releases of their products with improved features and extra functionalities. Software developers choose to keep rolling out new versions one after another, with later versions typically coming with added functionality. Most products start their life cycle with an initial version that includes enough features to make it useful for the customers. Later, they progress as new features are supplemented and existing features are enhanced through a series of releases. Hence, there is an utmost need to model multi-releases in software development process. Most of the work done in software reliability engineering pertains to single release software system with a plenty of development in the continuous models. In this paper, a multi-release discrete SRGM is proposed taking into account the significant difference between testing and operational profile of software development process. Different growth curves have been used to model the fault removal process (FRP) for these two phases and the effect of up-gradation on the subsequent releases is analyzed.

Authors also address the important issue involved in software development i.e., to determine optimal release time for the launch of the upgraded version. Excessive late entry may turn to significant loss of opportunity while early release of any software may delay its growth due to lack of potential users for latest expertise. In developing software with multiple releases, a major concern for the software development firms is to decide when to stop testing the current release and introduce a new version with additional features. Many release time problems with optimization criteria like cost minimization or profit maximization along with reliability or budgetary constraints etc. have been discussed in the literature (Kapur & Garg, 1991; Lin & Huang, 2006; Ping, Zhao, & Ke, L, 2010). But, these problems were studied for the software which is one-off type. In this paper, authors formulate an optimal release planning problem which minimizes total cost for the software with multiple releases by taking into consideration the cost incurred during testing and operational phase of the software.

LITERATURE REVIEW

The growing trend towards up-gradation of software’s has created lots of challenges to software developers to meet the changing user requirements for higher software quality (Kasurinen, Taipale, Vanhanen, & Smolander, 2012; Lapouchnian, Yu, & Deng, 2014). Some of the major challenges include requirements volatility, design process, quality issues (e.g., performance, usability, security, etc.), distributed software development, fast changing technology, efficient allocation of human resources to development tasks and limited budget (Mehrfard & Lhadj, 2011; Sauma, 2013). These challenges have prompted software developers to plan multiple releases for their product by adding new features and enhancements in the software at a very frequent rate, thus user is less likely to find himself in an obsolete version (Weerd, Mirandolle, & Brinkkemper, 2012). The phenomenon of adding features and functionalities to the existing products is called up-gradation. Due to the
Dynamic Data Mediation in Enterprise Application Integration Scenarios
www.igi-global.com/chapter/dynamic-data-mediation-enterprise-application/37931?camid=4v1a

Note on Assignment Algorithm with Easy Method of Drawing Lines to Cover All Zeros
www.igi-global.com/article/note-assignment-algorithm-easy-method/69180?camid=4v1a