Software evolution

Engineering

Introduction

Software engineering is arguably the most sophisticated invention that human beings have made in the modern times. There are many studies done on this field in an effort to improve the knowledge of computers and informatics for the betterment of business and the general global community. The following are two of those studies, their strengths, weaknesses, and comparisons.

Kaith and Rajlich (p. 77) highlight the necessity and requirements for software evolution and maintenance. They present a model roadmap for software evolution called the staged model. The staged model consists of four stages namely initial development, evolution stage, servicing stage and phase out stage. According to their article, each stage must preserve the software integrity and architecture developed in the initial stage, and that a previous stage must provide room for improvement in the next stage. The article also provides that reverse software engineering is the ultimate solution for software legacy problems that are likely to worsen in the future. The article ties business advancement with software evolution, saying that software evolution should be rapid to meet the needs of the rapidly evolving business environment.

Criticism

The main strength of this article lies in how it presents the staged model of software evolution. The model is practical and realistic because it addresses the basic need for software maintenance and improvement. The article appreciates that no software is completely sufficient or error proof, and hence the need for continuous debugging and improvement. The weakness of this article is that it solely ties software maintenance and development with an online business. However, software development is important for other areas that the article does not mention. The government, for instance, maintains classified information of national and international interest in highly-encrypted databases that should be ultimately secured. In fact, any software development needs to address the enhancement of this security, since a breach of it may lead to the collapse of business and other operations safeguarded by state.

Pamela et al (p. 2) demonstrates how graph topology can be useful in software evolution. Specifically, the graph method can be used to estimate bug severity, predict defect prone releases and capitalize on software refactoring endeavors. Two types of graphs for 11 open source programs are constructed. The first graph is the product level graph that captures coding and modulation. The other graph is the process level graph that captures developer collaboration efforts.

Criticism

The strengths of this project, unlike previous works on the same subject, is that it tracked graph metrics of multiple release software to detect peculiar pivotal moments in the software evolution and constant graph metrics that hold across the entire lifespan of a program. Another advantage of this project is that it used eleven open source systems that are reputable and commonly used like Firefox and VLC. The most notable weakness of this study is that it compared various graph metrics for only two or three software as shown on the graphs in the results section, rather than evaluating each metric for all the programs simultaneously.

Comparison

The first article, published in 2000, provides a model framework for future software maintenance. This article appreciated the need for renewing software to a better and more advanced version. The second and more recent article presents one method through which software maintenance can be performed. It discusses the use of graph metrics in enhancing release of less buggy versions of a program.

Conclusion

The two articles portray a chronological pattern that contrasts older and recent progress made in software evolution and maintenance. The first article describes a model formula for software evolution and maintenance while the second describes how graph metrics can be used to predict software versioning and development of error proof software.

Works Cited

Bennett, Keith H., and Václav T. Rajlich. " Software maintenance and evolution: a roadmap." Proceedings of the Conference on the Future of Software Engineering. ACM, 2000.

Bhattacharya, Pamela, et al. " Graph-based analysis and prediction for software evolution." Proceedings of the 34th International Conference on Software Engineering. IEEE Press, 2012.