Evolution Impact on Architecture Stability in Open-Source Projects

Mamdouh Alenezi, Prince Sultan University, Riyadh, Saudi Arabia
Fakhry Khellah, Prince Sultan University, Riyadh, Saudi Arabia

ABSTRACT

Software systems usually evolve constantly, which requires constant development and maintenance. Subsequently, the architecture of these systems tends to degrade with time. Therefore, stability is a key measure for evaluating an architecture. Open-source software systems are becoming progressively vital these days. Since open-source software systems are usually developed in a different management style, the quality of their architectures needs to be studied. ISO/IEC SQuaRe quality standard characterized stability as one of the sub-characteristics of maintainability. Unstable software architecture could cause the software to require high maintenance cost and effort. In this work, the authors propose a simple, yet efficient, technique that is based on carefully aggregating the package level stability in order to measure the change in the architecture level stability as the architecture evolution happens. The proposed method can be used to further study the cause behind the positive or negative architecture stability changes.

Keywords: Open Source, Package Structure, Software Evolution, Software Metrics, Software Quality, Stability

1. INTRODUCTION

Software evolution is the vigorous activities of software systems while they are improved and maintained over their lifespans (Lehman, 1980; Godfrey & German, 2013; Alenezi & Almustafa, 2015). Software systems change and evolve throughout their life cycle to accommodate new features and to improve their quality. Software needs to evolve in order to survive for a lengthy period. The changes that software undergo lie within corrective, preventive, adaptive and perfective maintenance that lead to software evolution. A major characteristic of software evolution is architecture evolution. While a specific system is evolving, its architecture is affected. In opposition, having a plan for how an architecture should evolve is a powerful mechanism to plan and guide software evolution.

Software Architecture is defined in the IEEE standards (ISO/IEC/IEEE, 2011) as “fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution”. One desired quality of the software architecture is stability. Stability is one of the maintainability characteristics of the ISO/IEC
SQuaRe quality standard (ISO/IEC, 2011). According to this standard, stability is defined as the degree to which the software product can avoid unexpected effects from modifications of the software. (ISO/IEC, 2011).

Abundant research studies addressed the software evolution of open-source systems, with more than one hundred research papers referenced in recent systematic literature reviews (Breivold, Chauhan, & Babar, 2010; Syeed, Hammouda, & Syat’a, 2013). Although there are abundant research studies that investigated the evolution of these softwares, very little effort targeted the architecture in general and no work addressed the architectural stability of these systems. Almost all stability related studies target the package level stability by using various metrics. In this work, we concentrate on the most frequently used metric for assessing the package stability which is defined as (Martin, 2003):

\[
\text{Instability} = \frac{C_e}{C_e + C_a}
\]  

The \( I \) metric for a package is defined as the ratio of efferent coupling \( C_e \) to total coupling \( C_e + C_a \) for the package. The \( C_a \) denotes the number of other packages that depend upon classes within the package (fan-in). It measures the incoming dependencies. The \( C_e \) denotes the number of other packages that the classes in the package depend upon (fan-out). It measures and counts the outgoing dependencies. The \( I \) metric indicates how a flexible a package is to change. The metric ranges from zero to one, one indicates a completely unstable package whereas zero indicates a completely stable package. Martin (Martin, 2003) suggests that some packages are easier to change than others. Easy to change packages should depend on less easy to change packages. Depending on packages make a package less stable, as changes affecting depended-upon packages propagate to the depending package. Having other packages depend on a certain package make that package more stable, as more effort is needed to merge changes with all dependent packages. The \( I \) metric shows how easy a package is to change.

Due to system evolution, the overall system stability is affected. This is due to mainly changes in the currently existing packages and the instability incurred due to the added new packages. It is very important for system architect to monitor the overall system instability due to various maintenance task such as adding and removing packages, hence, changing the packages dependency relationships. However, calculating an aggregate value that shows the overall system stability due to evolution has not been addressed in literature.

The main contribution of this work is two-fold. First, proposing a simple, yet efficient, technique that is based on carefully aggregating the package level stability in order to measure the change in the architecture level stability as the architecture evolution happens. Second, empirically applying the proposed method on two open source systems.

Empirical studies frequently rely on data sets. Since open source software projects are often developed by volunteers with no formalized development methods (Gyimothy, Fer- ene, & Siket, 2005), their process and resource data are often incomplete and unreliable. Consequently, to conduct and validate the empirical work of open source projects, product metrics only are exploited. By using open-source systems data, researchers will be able to assess and examine different hypotheses about the applicability of different software engineering methods. Open-source systems have been the source for most empirical studies since all their development data are publicly available.
The Effects of Customer Perceived Employee Support on Self-Efficacy and Behavioral Intentions: The Roles of Service Complexity and Choice Freedom
[www.igi-global.com/article/the-effects-of-customer-perceived-employee-support-on-self-efficacy-and-behavioral-intentions/145200?camid=4v1a](www.igi-global.com/article/the-effects-of-customer-perceived-employee-support-on-self-efficacy-and-behavioral-intentions/145200?camid=4v1a)

Loyalty Programme and Meta-Services: Insights from the Case of Airline Alliances
[www.igi-global.com/article/loyalty-programme-and-meta-services/173035?camid=4v1a](www.igi-global.com/article/loyalty-programme-and-meta-services/173035?camid=4v1a)