Measuring Structural Quality of Object-Oriented Softwares via Bug Propagation Analysis on Weighted Software Networks

Wei-Feng Pan, Bing Li, Ye-Yi Qin, Yu-Tao Ma, and Xiao-Yan Zhou

1 State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China
2 School of Computer, Wuhan University, Wuhan 430072, China
3 Complex Networks Research Center, Wuhan University, Wuhan 430072, China

E-mail: panweifeng1982@gmail.com; libing@sklse.org; yutaom@acm.org; qinyeyi2005@126.com; zhou0420@tom.com

Received July 15, 2009; revised May 17, 2010.

Abstract The quality of a software system is partially determined by its structure (topological structure), so the need to quantitatively analyze the quality of the structure has become eminent. In this paper a novel metric called software quality of structure (SQoS) is presented for quantitatively measuring the structural quality of object-oriented (OO) softwares via bug propagation analysis on weighted software networks (WSNs). First, the software systems are modeled as a WSN, weighted class dependency network (WCDN), in which classes are nodes and the interaction between every pair of classes if any is a directed edge with a weight indicating the probability that a bug in one class will propagate to the other. Then we analyze the bug propagation process in the WCDN together with the bug proneness of each class, and based on this, a metric (SQoS) to measure the structural quality of OO softwares as a whole is developed. The approach is evaluated in two case studies on open source Java programs using different software structures (one employs design patterns and the other does not) for the same OO software. The results of the case studies validate the effectiveness of the proposed metric. The approach is fully automated by a tool written in Java.

Keywords bug propagation, design pattern, object-oriented (OO) software, software network, structural quality

1 Introduction

Object-oriented (OO) has been the most widely used development paradigm since the early 1990’s. And there are a large number of open source object-oriented (OSOO) software systems in free software ecosystems such as Sourceforge[1] and Freshmeat[2]. Many of them have equivalent or overlapping functionalities. The lack of objective evaluation criteria makes the choice of the best one from these candidate OSOO software systems difficult. Furthermore, as the OSOO software systems have considerable economic impact, and make great inroads into the mission-critical or life-critical real-world applications, many organizations would like to have object measures regarding the quality of the software products[3]. So there is a great need for some objective metrics to quantify the quality of OSOO software systems.

Software structure design explicates the structure of the software in terms of software components and interactions among them. Especially, in OO software systems, it describes methods, attributes, classes, packages, etc., and their interactions. With the increase of the complexity of software systems, the overall structure of the system is becoming more and more complicated, making the software structure become one of the most important factors that influences the quality of the final software products[4]. So the need to quantitatively analyze the quality of the structure has become eminent[5].

In recent years, researchers in the field of statistical physics and complex system used complex software networks (hereafter, software networks) to represent software systems by taking software components, such
as methods, classes and packages, as nodes and their interactions as edges\(^6\). It provides us a new way to study complex software systems. And the research interests are mainly involved in discovering the shared topological properties of software networks, the evolution mechanisms of software networks and the metrics for evaluating complexity of software networks. [7] gives a detailed review of the research in this new field.

But to the best of our knowledge, quantitative studies of software quality from the perspective of software networks (i.e., to quantify the structural quality of software networks) are very scarce, and the assumptions they are based on cannot meet the practice to a certain degree. Considering the defects of the existing methodologies, in this paper we propose modelling OO software systems at class level using a weighted software network (WSN), weighted class dependency network (WCDN), which is built on the details of the features (i.e., methods and attributes) and their interactions in the specific OO software system. And then, we present a new metric, software quality of structure (S\(\text{SQoS}\)), to measure the structural quality of OO softwares by analyzing the bug propagation process in WCDN and the bug proneness of each class. We test our approach against two case studies on open source Java programs, each of which has two versions (one employs design patterns and the other does not). The results of the case studies validate the effectiveness of S\(\text{SQoS}\). The approach is fully automated by a tool written in Java.

The rest of this paper is organized as follows. Section 2 contains a brief summary of the related work. Section 3 describes our approach in detail. Section 4 presents the results of two case studies conducted on two open source case studies. In Section 5 a software tool that has been developed to automate the proposed approach is briefly described. The limitations and future work of our research are mentioned in Section 6. And we conclude the paper in Section 7.

2 Related Work

Let us brief the researches on software structural quality measurement first, and then detail some research work from the perspective of software networks.

In [8], Alan MacCormack et al. adopt design structure matrices (DSMs) to represent the software networks at the source file level, and introduce change cost to measure the average influence of components on the whole system. Basically, a software system should keep the change cost as low as possible, i.e., the influence of any performed change should be limited to a range as small as possible. However, it assumed that all files have the same probability that they can be changed, and that a change in one file will definitely propagate to other files that point to it directly and indirectly in software networks. This may not meet the practice.

In [9], Dannien Challet et al. propose a metric called failure propagation basin to study the bug propagation in function call graphs and package dependence graph. The failure propagation basin measures the potential influenced nodes caused by a faulty node (contains a bug) and it is defined as the number of nodes that point to the faulty node directly or indirectly in software networks. They calculated the failure propagation basin of each node, and studied the distribution curves of the size of failure propagation basins. They, however, neither take into account the significance of the attributes in bug propagation nor propose a metric to characterize the structural quality of the software systems as a whole.

In our preliminary work\(^{10}\), we introduced an efficient statistical measure, called average propagation ratio, to characterize the structural quality of general complex software networks at the granularity level of class. And several representative real-world complex software networks were analyzed using average propagation ratio. However, average propagation ration also assumed that all classes have the same probability that they may be changed or may be faulty (contains a bug), and that a change or bug in one class will definitely propagate to other classes that point to it directly and indirectly in software networks, which may not meet the practice.

3 The Approach

In the previous researches as we talked in Section 2, people always make the following two assumptions: 1) the change or bug in one software component such as a file or a class will definitely propagate to other components that point to the changed or faulty node directly or indirectly in software networks; and 2) all software components, such as classes, and files, have the same probability that they may be changed or may be faulty. But these two assumptions may sometimes not meet the practice. The rationale is twofold.

1) In OO software systems, a class always contains many attributes and methods. We treat a class as a faulty one if it contains at least one bug. Similarly, the attributes and methods of another class depending on the faulty class do not all link to the faulty attributes or methods directly or indirectly in it.

See Fig.1, class X is composed of three methods (\(b()\), \(c()\), \(d()\)) and one attribute (\(a\)), and class Y is composed of two methods (\(e()\) and \(f()\)). So if any attribute or method of class X is faulty, class X will be viewed as faulty. In previous work, people all think the bug in class X will definitely propagate to class Y, for the