Physica A 392 (2013) 2025–2037

Contents lists available at SciVerse ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

A modular attachment mechanism for software network evolution

Hui Li *, Hai Zhao, Wei Cai, Jiu-Qiang Xu, Jun Ai

College of Information Science and Engineering, Northeastern University, Shenyang, 110819, China

ARTICLE INFO

Article history:
Received 18 August 2012
Received in revised form 10 January 2013
Available online 21 January 2013

Keywords:
Software networks
Evolutionary mechanisms
Modular attachment
Asymmetric probabilities
Power-law

ABSTRACT

A modular attachment mechanism of software network evolution is presented in this paper. Compared with the previous models, our treatment of object-oriented software system as a network of modularity is inherently more realistic. To acquire incoming and outgoing links in directed networks when new nodes attach to the existing network, a new definition of asymmetric probabilities is given. Based on this, modular attachment instead of single node attachment in the previous models is then adopted. The proposed mechanism is demonstrated to be able to generate networks with features of power-law, small-world, and modularity, which represents more realistic properties of actual software networks. This work therefore contributes to a more accurate understanding of the evolutionary mechanism of software systems. What is more, explorations of the effects of various software development principles on the structure of software systems have been carried out, which are expected to be beneficial to the software engineering practices.

1. Introduction

Many large systems natural and man-made, such as WWW [1,2], science collaboration [3–5], social networks [6,7] and transportation networks [8], have shown global statistical and evolutionary characteristics. Object-oriented software systems represent one of the most diverse and sophisticated man-made systems; however, only little is known about the actual structure and forming mechanism of large software systems. Although it is widely acknowledged that software evolution depends on its architecture, the cause-and-effect relationships between design practices and evolution outcomes have not been thoroughly investigated [9]. As a consequence, software networks research on analyzing and modeling software systems as complex networks [10] can afford us deep understanding about the formation, evolution of code-based software structures and the processes governing the development of software systems.

In recent years, many empirical studies have been greatly helpful for understanding the software systems. They have uncovered that software networks, extracted from various software systems, follow power-law degree distributions [11,12,10,13–17], represent small-world properties [10,18,19], exhibit community phenomena [10,20], and show some other complex behavior characteristics [11,21–26]. Furthermore, many works have analyzed software systems in different network scales by employing a variety of software networks, such as software motifs [27,28], package, class and method collaboration graphs [21,29]. Obviously above-mentioned results have revealed a structure of clustering [19] and modularity [20] in software networks [10]. Based on these works, research on software network evolution is emerging and has attracted wide-spread attention. Some empirical studies [21,22,29–33] have identified and formulated some network structural evolution laws which are obeyed by most of software systems, and these laws can be universally applied in practices.

* Corresponding author. Tel.: +86 13609826177.
E-mail address: lih2002@126.com (H. Li).

0378-4371/$ – see front matter
doi:10.1016/j.physa.2013.01.035

On another research front, a few models of software network evolution, which describe the evolutionary mechanisms from different perspectives, have been proposed. For instance, Myers [10] has proposed a model based on refactoring processes. A model based on the growth of software patterns has been presented by He et al. [34]. Valverde et al. [27] have put forward a model based on node duplication plus edges rewiring. Some other models have also simulated the process of software network evolution from different perspectives (such as Refs. [26,35]). As one of the most popular and accepted model for simulating scale-free network evolution, the Barabási–Albert (BA) model [11] is based on two mechanisms: incremental growth and linear preferential attachment. On the basis of these, a novel nonlinear preferential attachment mechanism has been presented by Krapivsky et al. [36]. Dorogovtsev and Mendes [37] have also proposed a model relying on both the degree of the existing node and its age. Moreover, Zheng et al. [38] have proposed the Degree and Age Dependent Adjustable Evolution (DAAE) model based on the above models for modeling software network formation.

While the above-mentioned models represent good performance on various aspects when modeling software network evolution, there still exist some limitations of them, and the evolutionary mechanisms of software networks are also unclear. On the one hand, aforementioned models are usually lack of modularity. In fact, software systems represent highly functional modularity [10] in real-world software engineering, and this phenomenon has been confirmed in the community structure of software networks [20]. As Fortuna et al. [30] have revealed, the modularity is increasing as software network evolves. On the other hand, most of the previous models undertake undirected network treatment in software networks as some complex networks such as Internet, social networks. Actually, software networks are inherently directed. That is because dependent relations between basic units in software systems are unidirectional.

The above considerations motivate the study in this paper. A mechanism for evolution of software systems from a new angle of view is proposed. In order to represent connectivity of the constituent in and out connections for classes, we extract directed networks from software systems. Then, modular attachment instead of single node attachment obtained in the previous models [10,34–38] is put forward, which is able to promote the emergence of structural modules within software networks naturally. We also obtain asymmetric probabilities for incoming and outgoing preferential attachment, in order to represent unidirectional dependent relations between classes. This mechanism of software network formation may provide a new angle for observing the organization of software systems.

The rest of this paper is structured as follows. After describing modular attachment model in terms of structure and evolutionary mechanism in Section 2, we compare the simulations of this model with actual software networks and simulations of two previous models in Section 3. In Section 4, we discuss implications of some software design principles for the structure of software networks. In Section 5, the conclusion of this paper is presented and possible future works are proposed.

2. The mechanism of software network evolution

2.1. Software networks

A software system is composed of many interacting units (e.g., classes, components and subsystems) and the collaborations among them directly reflect the design, coding, and execution of software.

Particularly, software interactions imply dependency relationships, in that some basic units need others in order to carry out their assigned task. Thus the intention of software design and development is to "construct an optimal or near-optimal system of dependency relationships, whereby core elements are reused in different contexts to perform recurring fundamental tasks, with minimally constraining specializations added in higher functional layers in order to build upon or combine those fundamental tasks" [10]. We can therefore define software network as $N = \{V, E\}$, where $V = \{v_1, v_2, \ldots\}$ is the set of nodes which denote classes, interfaces and struts in software systems, and $E = \{e_{i,j}, \ldots\}$ is the set of edges in which $e_{i,j} : v_i \to v_j$ represents dependency relationship from node v_i to v_j.

Specifically, class dependencies include two types of relations between classes: inheritance represents "is a" relation and association represents "has a" relation. The direction of dependency relations reflects the flow of control in a software system: an edge in a class collaboration graph is directed from class B to class A if B makes reference to A in its definition (either through inheritance or association). In other words, the direction of the edges in software networks is determined in the following manner: (a) it will establish an outgoing edge to an existing node if the corresponding class inherits that class or use an instance of that class; (b) it will establish an incoming edge to an existing node if the corresponding class is inherited by that class or its instance is used by that class. Repeated connections are not considered in the following analysis.

Each node is characterized by its degree k_i, which is the number of edges attached to it. Additionally, in-degree k_i^{in} and the out-degree k_i^{out} for node v_i are defined as the number of links entering the node and the number of links exiting the node, respectively.

We choose 8 versions of Eclipse (version 2.0.1, 2.1.3, 3.0.1, 3.1.1, 3.2.2, 3.3.1.1, 3.4.2, 3.6.2) in time serial as empirical cases in this paper. Eclipse is well-known and has large network size, and the number of nodes is from 6526 in version 2.0.1 to 21975 in version 3.6.2.

2.2. Software network growth

In terms of the software development process, iterative and incremental development is widely applied under the banner of agile development and lowers the risks and costs [39].