Physica A 390 (2011) 2968–2975

Contents lists available at ScienceDirect

Physica A

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

Community structure of complex software systems: Analysis and applications

Lovro Šubelj *, Marko Bajec

Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia

ARTICLE INFO

Article history:
Received 10 August 2010
Received in revised form 1 November 2010
Available online 12 April 2011

Keywords:
Community structure
Complex networks
Software systems

ABSTRACT

Due to notable discoveries in the fast evolving field of complex networks, recent research in software engineering has also focused on representing software systems with networks. Previous work has observed that these networks follow scale-free degree distributions and reveal small-world phenomena, while we here explore another property commonly found in different complex networks, i.e. community structure. We adopt class dependency networks, where nodes represent software classes and edges represent dependencies among them, and show that these networks reveal a significant community structure, characterized by similar properties as observed in other complex networks. However, although intuitive and anticipated by different phenomena, identified communities do not exactly correspond to software packages. We empirically confirm our observations on several networks constructed from *Java* and various third party libraries, and propose different applications of community detection to software engineering.

1. Introduction

Analysis of complex real-world networks has led to some significant discoveries in recent years. The research community has revealed several common properties of different real-world networks [1–3], including various social, biological, Internet, software and other networks. These properties provide an important insight into the function and structure of general complex networks [4,5]. Moreover, they allow for better comprehension of the underlying real-world systems and thus give prominent grounds for future research in a wide variety of different fields.

In the field of software engineering, network analysis has just recently been adopted to acquire better comprehension of the complex software systems [6–10]. Nowadays, software represents one of the most diverse and sophisticated human made systems; however, only little is known about the actual structure and quantitative properties of (large) software systems. Cai and Yin [9] have denoted this dilemma as the *software law problem*, which represents an effort towards identifying and formulating physics-like laws, obeyed by (most) software systems, that could later be applied in practice. However, the main objective of the software law problem is to investigate what software looks like.

In the context of employing complex networks analysis, the research community has already made several discoveries over the past years. In particular, different authors have observed that networks, constructed from various software systems, follow *scale-free* [2] (i.e. power-law) degree distributions and reveal *small-world* [1] phenomena [11,6]. We proceed this work by exploring another property commonly found in other real-world networks, i.e. *community structure* [3]. The term denotes the occurrence of local structural modules (*communities*) that are groups of nodes densely connected within and only loosely connected with the rest of the network. Communities play crucial roles in many real-world systems [12,5], however, the

* Corresponding author.
E-mail address: lovro.subelj@fri.uni-lj.si (L. Šubelj).

0378-4371/$ – see front matter
doi:10.1016/j.physa.2011.03.036

community structure of complex software system networks has not yet been thoroughly investigated. For a comprehensive survey on network community structure see Refs. [13,14].

The main contributions of our work are as follows. We adopt *class dependency networks*, where nodes represent software classes and edges represent dependencies among them, and show that these networks reveal a significant community structure, with similar properties as observed for other complex networks. We also note that a network, representing a core software library, exhibits less significant community structure. Furthermore, we prove that, although intuitive and anticipated by different phenomena, revealed communities do not (completely) correspond to software packages. Thus, we demonstrate how community detection can be employed to obtain highly modular software packages that still relate to the original packages.

The rest of the article is structured as follows. First, in Section 2, we briefly present relevant related work and emphasize the novelty of our research. Next, Section 3 introduces employed class dependency networks. In Section 4 we present an empirical evaluation of the community structure of class dependency networks, and propose possible applications to software engineering. Finally, in Section 5, we give final conclusions and identify areas of further research.

2. Related work

Although software systems have already been investigated over the last 30 years [15], the research community has only recently begun to employ network analysis to gain better comprehension of complex software [6–8,16,9,10,17]. As mentioned above, different authors have observed that networks, constructed from software systems, follow scale-free degree distributions [11,6,18,19] and exhibit small-world properties [6,20,21]. Software networks thus reveal common behavior, similar to that observed in other complex networks [4,3]. Furthermore, authors have also identified several different phenomena (e.g. software optimization and tinkering) that might govern such complex behavior [11,22,23,21,24], when the analysis of *clustering* [1] has also revealed a hierarchical structure in software networks [6].

On the other hand, the community structure of software networks has not yet been thoroughly investigated. Several authors have already discussed the notion of communities in the context of software systems [6,22,25,21,26,16], however, no general empirical analysis and formal discussion was ever conducted (to the best of our knowledge). Still, authors have observed different phenomena that could promote the emergence of community structure in software networks [25,16], and have discussed possible applications within software engineering and other sciences [6,16]. Otherwise, several models (processes) of network evolution have already been proposed to explain the emergence of local structural modules in different networks. These include increase of stability [27], network fluctuations [28], goal variation [29], opinion formation [30], constraint optimization [31] and others [32]. However, the origin of community structure in software networks remains unclear.

In a wider context of software network analysis, authors have discovered reoccurring *motifs* (i.e. small subgraphs) in software networks [8], similar to those found in genetic and neural networks. In Ref. [7] a network model is presented, which explains observed asymmetries in degree distributions of software networks (see, e.g., Ref. [6]). Furthermore, different random-walk based measures have been proposed to measure key (i.e. most influential) software classes and packages [33,10]. The researchers have also investigated connectedness, robustness and patterns within software networks [6,26]. Just recently software systems were also treated as evolving complex networks [9]. For a more general discussion on software networks see Ref. [34].

3. Class dependency networks

Previous research on the analysis of software systems has employed a variety of different types of software networks (i.e. graphs). In particular, *package*, *class* and *method collaboration graphs* [6,18], *subrutine call graphs* [6], *software architecture* [26] and *software mirror graphs* [9], *software architecture maps* [11], *inter-package dependency networks* [20] and many others [6,21,36]. The networks primarily divide whether they are constructed from source code, byte code or software execution traces, and due to the level of software architecture they represent. However, as discussed in Section 2, most of these networks share some common characteristics.

For the purpose of this research we introduce *class dependency networks* (Fig. 1). Here an object-oriented software is represented by an undirected multi-graph $G(N, E)$, where N is the set of nodes and E is the set of edges. Graph G is constructed from software source code in the following manner. Each software class c is represented by a node $n_c \in N$, when edge $\{n_{c_1}, n_{c_2}\} \in E$ represents a *dependency* between classes c_1 and c_2. Dependencies are of four types, namely, *inheritance* (class c_2 inherits or implements class c_1), *field* (c_2 contains a field of type c_1), *parameter* (c_2 contains a method that takes type c_1 as a parameter) and *return* (c_1 contains a method that returns type c_2).

Note that class dependency networks are constructed merely from the header information of the classes, and their methods and fields. As this information is commonly determined by a group of developers, prior to the actual software development, it is less influenced by the subjective nature of each particular developer. Hence, the networks more adequately represent the (intended) structure of some particular software, still, some relevant information might thus be discarded.

An example of class dependency network is shown in Fig. 1. The network reveals rather strong community structure, furthermore, the communities also roughly coincide with the actual software packages. However, as will be shown in