Stochastic Models for Evaluating Software Modularization Recovery Algorithms

Rodrigo Souza
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Brazil
rodrigorgs@gmail.com

***Abstract*—Software modularization recovery algorithms help to understand how software systems decompose into modules, but they**

Software modularization recovery algorithms... bla bla

***Keywords*-reverse engineering; software modularization; empirical evaluation; complex networks;**

I. Abstract

Software modularization recovery algorithms automatically recognize a system's modular structure by analyzing its implementation. Due to the lack of well document software systems, though, the issue of testing these algorithms is still underexplored, PREJUDICANDO both their ADOCAO in the industry and the development of better algorithms. We propose to rely on software models to produce arbitrarily large test sets. In this paper we consider three such models and analyze how similar the artifacts they produce are from artifacts from real software systems.

II. Introduction

Development of large-scale software systems is a challenge.

A key to success is the ability to decompose a system into weakly-coupled modules, so each module can be developed by a distinct team. Failing to do so results in duplicated code, non-parallelism, one's work impacting another's work etc.

The ability do modularize depends decisively on a vast knowledge about the system, how its different parts interact to accomplish the system's goal.

Unfortunately, in the case of legacy systems, such knowledge isn't available. Depending on its size, it might take months to understand the system so well as to find a good modularization.

POR ISSO SURGIRAM software modularization recovery algorithms, also known as software clustering algorithms or software architecture recovery algorithms. In its most common flavor, these algorithms analyze the dependencies between implementation components, such as classes, and then group them into modules such as there are few dependencies between classes in distinct modules.

Software modularization recovery algorithms can, therefore, do in minutes what a person would spend weeks or months. The question is: are the found modularizations good? Are they similar to what a person would find? To answer this question it's essencial to perform empirical evaluations envolving systems with known reference modularizations.

The empirical evaluations consist of selecting a collection of systems with known reference modularizations and then applying the algorithms to the systems. The modularizations found by the algorithms are then compared to the reference decompositions by a metric such as MoJo CITE or PrecisionRecall CITE.

Unfortunately there are few systems with known reference modularizations and, because to obtain reference modularizations is costly, there are few empirical studies, and most of them consider a couple of small and medium systems.

We therefore propose to use synthetic, i.e., computer-generated, software dependency networks, to evaluate software modularization recovery algorithms. These networks are generated by parametrizable models and have an embedded reference modularizations. The goal of an algorithm is, thus, to find modularizations that are similar to the reference modularization embedded in the network. With this approach we can CONTAR COM a large volume of test data that is composed of networks of different sizes and controllable characteristics.

Of course the success of this approach depends on the realism of the synthetic networks, ie, how well they resemble networks extracted from real software systems. In this paper we study three models and show that all of them are, by means of a careful parameter choosing, capable of producing realistic software networks.

The remaining sections are organized as .... Section 2, ...

III. Software Networks

directed graph, (un)weighted

IV. Complex Networks

Complex network theory found many scale-free networks

Software dependency networks are scale free. CITE Valverde, Myers

Scale free means ... $N(k) \sim k^{-\gamma}$

V. Models

Many scale-free models have been proposed. Only a few, though, produce modular networks.

A. LF

Directed weighted networks with overlapping modules.

B. BCR plus

We propose an extension to BCR model... Growth model.

C. CGW

Accounts for the removal of edges. Growth model.

VI. Experimental Setup

We want to show investigate if the models can produce networks that resemble software networks. We know that they share with software networks the scale-free property. This is not enough, since many real networks share this property. So we looked for a method to differentiate between software networks and other networks.

What we are looking for, after all, is an oracle that accepts software networks and rejects non-software networks. If the oracle has these two properties, we can be confident that it'll accept only synthetic networks that resemble software networks.

In a recent work, Milo et al. proposed the study of triads in order to characterize different classes of networks. We thus follow their work here.

Triads are...

Figure 1a show triads for a software system... Figure 2a show triads for network from domain X.

We've collected 65 systems written in Java and X networks from many domains, such as sociology, biology, technology and linguistics. Table 1 ...

We then used Pearson's correlation coefficient as a similarity measure between two networks. For each software network, We then computed the average correlation coefficient (ACC) to the other software networks. We've observed that among software networks the ACC is X +- Y.

We then computed, for each non-software network, the ACC to the 65 software networks. By the 3-sigma rule, we use X as the threshold for realistic software networks: networks whose ACC is below this threshold are rejected.

The oracle has X precision and X recall...

We generated networks with many combinations of parameters...

BCR: 5 different architectures, p, q, r in (0.0, 0.2, 0.4, 0.6, 0.8, 1.0), with p+q+r = 1 and p + q ¿ 0. ... Total: X networks

LR: ...

CGW: ...

VII. Experimental Results

All models produce networks that resemble software networks.

For some parameters, though, the networks are not realistic.

We cannot blame one single parameter for the non-realism.

VIII. Conclusion and Future Work

Bla-bla

Future work: to apply algorithms to the networks and compare the results with results found in the literature.

IX. Conclusion

The conclusion goes here. this is more of the conclusion

Acknowledgment

Dalton, Jorge, Christina, Garcia, Charles, Fabiola, Italo, Roberto, Jemerson