2012 28th IEEE International Conference on Software Maintenance (ICSM)

SEANets: Software Evolution Analysis with Networks

Theodore Chaikalis, George Melas and Alexander Chatzigeorgiou
Department of Applied Informatics
University of Macedonia
Thessaloniki, Greece
Email: {chaikalis, melas, achat}@uom.gr

***Abstract*—Evolving software systems can be systematically studied by treating them as networks and employing ideas and techniques from the field of Social Network Analysis. SEANets is an Eclipse plugin that allows the analysis of multiple software versions and the extraction and visualization of network properties required to investigate evolutionary trends of the underlying system.**

Keywords-Software evolution; network analysis; object-oriented design;

I. INTRODUCTION

Social networks have experienced a phenomenal growth in an extremely short period of time [1] and their expansion has been followed by countless studies on their structural properties. However, as it has been recently acknowledged, social networks are not static and therefore their evolution has attracted the interest of the relevant community. These efforts have resulted in elegant models of network growth shedding light into the mechanisms that underlie network evolution.

Object-oriented software systems can be effectively represented as graphs where nodes correspond to elements of the design (such as classes or methods) and edges to any kind of relation among them [2]. Given that most software systems are multi-version projects, their evolution is worth investigating by observing how fundamental network properties vary with time.

In this paper we present SEANets, which is an easy-to-use Eclipse plug-in associated with a public repository, for the automated analysis of multiple versions of any object-oriented software written in Java. SEANets allows the extraction and storage of network properties for all examined versions as well as the generation of various reports and charts focusing on the evolution of these properties. The tool's output is available through a Web page. The use of a public repository encourages the collection of data for several software systems which will be available to the relevant community for further analysis.

The study of network evolution in a social setting might be useful to understand the nature of the interactions among people. Software evolution analysis on the other hand, aims at interpreting macroscopic network phenomena and trends by analyzing graph properties at the node level [3], [4]. The ultimate goal is to relate evolutionary trends with qualitative properties of the examined software.

While both the terms *network* and *graph* refer to a collection of objects in which some pairs are connected by links [5], we employ the term *network* to emphasize the fact that techniques are borrowed from the field of Social Network Analysis.

II. EXISTING TOOLS

Various noteworthy tools have been developed by research teams for large (social or other) network analysis, featuring numerical calculations and visual representation of the corresponding data. Wikipedia lists over 70 products under the term social network analysis tools and libraries. A number of them like UCINet [6], Gephi [7], Pajek [8] and GUESS [9] are standalone programs for network visualization and metrics computation while others are software libraries offering customizable capabilities like SNAP [10] for C++ and NetworkX [11] for Python. Notable systems like NetMiner [12], igraph [13] and Cytoscape [14] have focused in the efficient graphical representation of the underlying networks while others like SNAP focus in the analysis of ultra large-scale connected components employing techniques based on sampling.

However, all of these tools do not focus on software and they assume that a dataset containing an edge list, a node list or a full matrix representation is already available. Consequently, in order to analyze software one has to build his own parser to extract the software representation and port the results to the corresponding tool.

On the other hand several tools have been developed to assist software maintenance by visualizing various aspects of software evolution; however, to the best of our knowledge these tools do not treat the software system as a network. These tools essentially offer a timeline for the visualization of software artifacts, such as project hierarchies [15] or metric values [16].

The proposed tool focuses in software evolution analysis allowing the user to map multiple versions of a Java based object-oriented system to a set of directed graphs. Several interesting properties can be calculated and represented visually following a *one-click approach* manipulating an easy to use web interface.

III. AVAILABLE ANALYSES AND REPORTS

In the context of SEANets, object-oriented software projects are regarded as directed graphs where nodes correspond to classes and an edge from a class *A* to a class *B* indicates the presence of a method call following the *Law of Demeter* [17]. Such a "friendship" between a source and a target class can be formed by references to the target class which are either attributes, local variables to which instances of the target class are assigned, method parameters and return types of the target class type.

This work has been partially funded by the Research Committee of the University of Macedonia, Greece

978-1-4673-2312-3/12/$31.00 ©2012 IEEE

Obviously, other kinds of class relations could also have been taken into account; however, the selected aspect emphasizes the notion of a class as a service provider to other clients and the communication among them as interactions to achieve the overall system requirements.

A. Basic Network Properties

In any attempt to study the evolution of a network one should be able to observe the size properties, such as the number of nodes, edges and diameter. SEANets provides line charts concerning the evolution of these metrics as well as a combined chart where all information is displayed on the same plot (Fig. 1). In Fig. 1 (results are shown for project FreeCol, which is an open-source strategy game [18]), each circle corresponds to an examined software version and the coordinates of the center of each circle indicate the number of nodes in that version (*x*-axis) and the number of edges (*y*-axis). The diameter of each circle is proportional to the actual graph diameter. By mouse hop over it is possible to see the exact numbers for these properties.

Such a plot indicates whether the density of the graph increases or not and whether a shrinking or expanding diameter model should be assumed for the underlying system. For example, Leskovec et al. [19] claim that social networks are governed by a shrinking diameter phenomenon according to which nodes get closer to each other over time. Initial evidence from software projects however, does not seem to validate this observation, i.e. the diameter of a software network appears to fluctuate over time. Further experiments are required in order to discover the root cause for this behavior.

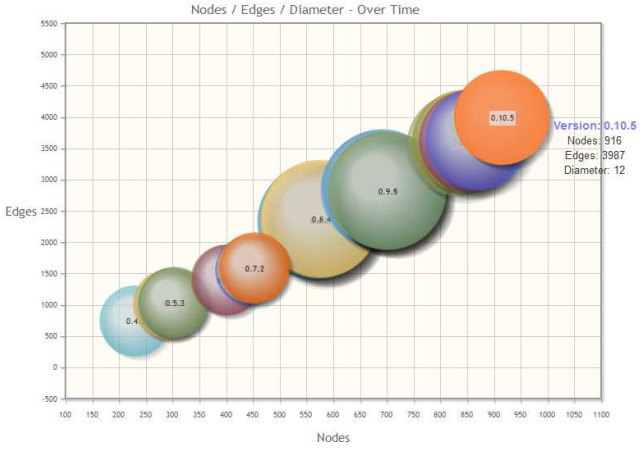

Fig.1. Evolution of Nodes/Edges/Diameter over time

B. Evolution of Edges

One of the major goals in studying the evolution of a network, regardless of the domain, is to gain insight into the mechanisms that govern their growth and if possible to determine the model that explains the corresponding evolutionary trends [20], [21]. Such a model can serve as a predictor of future evolution and possibly highlight problems in the design, such as the generation of God classes (authority or hub nodes) in future versions of a software system.

The starting point for building such models is the knowledge of how links among nodes are formed over time. To this end, SEANets provides line charts illustrating the evolution of all kinds of edges that might be present in a network: edges between existing nodes, edges from existing to new nodes, edges from new to existing nodes, edges between new nodes and edges that have been deleted in a new version. Fig. 2 shows the number of edges departing from new and reaching existing nodes for all examined versions of project FreeCol. As it can be observed there is a pattern of periodical spikes where a relatively large number of such edges are added, followed by versions with very limited edges of this type. Further analysis could possibly relate the spikes to major releases of the project.

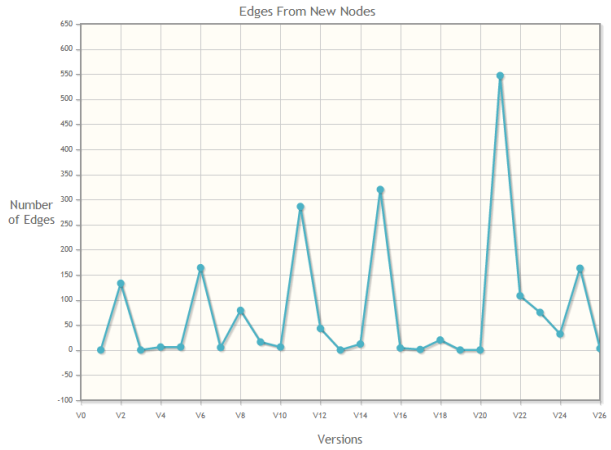

Fig.2. Evolution of edges from New to Existing Nodes over time

C. Additional Network Properties

Even a detailed record of how often edges are created or deleted is generally not sufficient to study the evolution of a network, since the creation of edges depends on numerous parameters such as the degree of the source and destination node, the age of the corresponding nodes, how far edges reach in the network, etc. In order to build more detailed growth models one needs to know detailed distributions of graph properties, such as the number of edges departing and flowing into each node versus the node age, the degree distribution, hop plots (illustrating the number of class pairs which are *h* hops apart) and so on. Fig. 3 illustrates the hop plot for the last examined version of FreeCol. The peak for example, indicates that more than 100,000 class pairs in that version are 3 hops apart (meaning that by traversing directed edges a path of length 3 is required to reach one class from the other). Moreover, it is evident that the graph diameter for that version is 12, since there is a (small) number of class pairs which are 12 hops apart (which is also visible in the diameter of the last version, in Fig. 1). This kind of plots illustrate visually whether the small world phenomenon is present [22] according to which any two nodes of a network can be linked by a path that is relatively short in length.

A second example of this type of information is a scatter plot which visualizes the relation of the clustering coefficient to the total degree of each node. The clustering coefficient of a node is equal to the probability that two random friends of this node are friends to each other [5]. Fig. 4 illustrates the

corresponding plot for the last examined version of FreeCol. As it is reasonable to expect, classes with a large total (in + out) degree have a low clustering coefficient, since it is unlikely that their numerous friends are also friends to each other. On the other hand, classes with very few friends have a significantly higher clustering coefficient, indicating that their neighbors are also exchanging messages. (The solid line corresponds to the trend captured by exponential regression.)

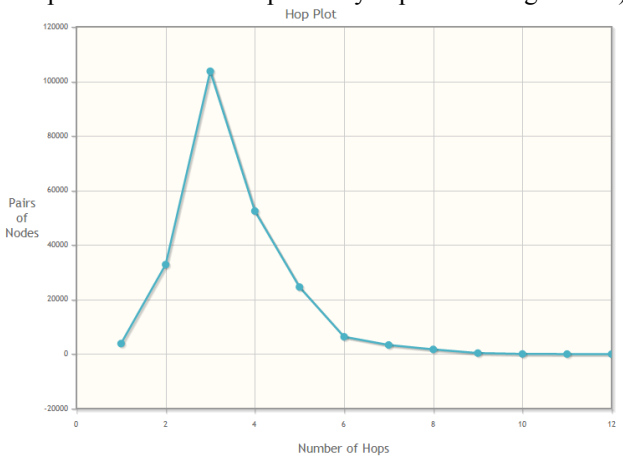

Fig.3. Number of class pairs which are *h* hops apart

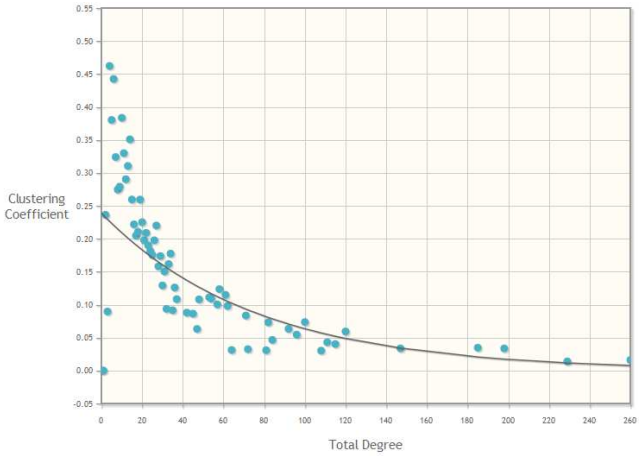

Fig.4. Clustering coefficient vs. Total Degree of each node

D. Network Visualization

Despite the fact that network visualization is not a prerequisite in order to build an accurate network evolution model, a visual representation of the network structure might be sometimes helpful to easily identify particular properties, such as hubs or authorities in the graph [5]. SEANets employs the Infovis Javascript library [23], which implements the Fruchterman-Reingold Force Directed algorithm [24] for displaying the network's connected components. A sample visualization is shown in Fig. 5. The visualization is interactive in the sense that the user can select and move around any node of the network.

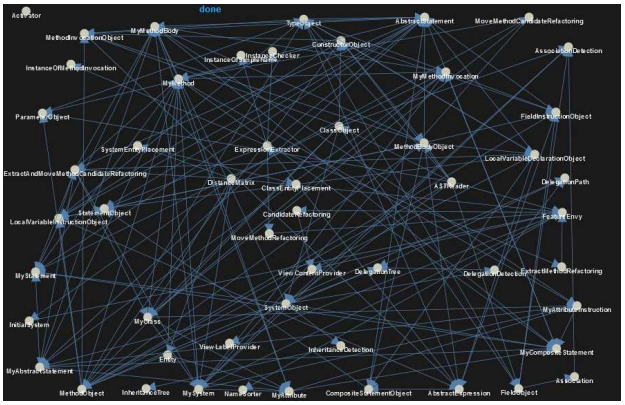

Fig.5. Network Visualization

IV. Tool Architecture

Figure 6 depicts graphically the main components and data flows involved in SEANets. An Eclipse plugin written in Java parses and analyzes the source code of multiple versions of the software project under study, employs the Abstract Syntax Tree to extract the graph corresponding to each version and stores the results in a remote database. The system default browser opened within the Eclipse workbench retrieves the selected results from the database and employs JavaScript to generate various reports and charts.

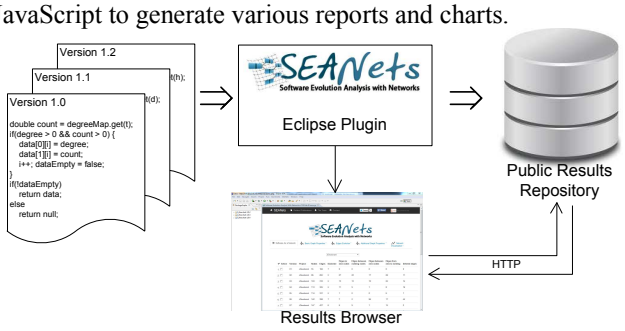

Fig.6. Graphical Overview of the Tool's Architecture

SEANets web reports are built upon HTML5 technologies that today leverage the user experience by offering responsive interface components for best view on large and small screens. The PHP backend system queries a MySQL database via Ajax calls triggered when the user selects various network properties. The retrieved data are visualized by means of JavaScript libraries, thus offering the user an easy-to-use, powerful tool, managed by a thin client and accessed from any web browser.

V. Typical Usage Scenario

The primary scenario assumes that the user is interested in analyzing the evolution of an object-oriented software project for which various versions are available as source code in the local workspace. The user follows the following steps:

1. The user imports into Eclipse the versions of the project that he wishes to analyze.
2. The user clicks the "Project Evolution Analysis" button in order to initiate the analysis process.

3. After the successful completion of the analysis, the results will be saved automatically into the Database and the Results Browser will appear.
4. The user selects his project from the Project Selector. The overview table appears.
5. The user selects a single or multiple (depending on the desired analysis) versions from the overview table.
6. The system displays the desired chart. (By mouse hop over the user can see additional data on several charts).

In case the user has already analyzed a given system the results are already stored in the database. Thus, access to the results is possible without the need to re-analyze the project versions, by simply selecting the desired system from the project selector.

VI. Missing Features and Future Work

Several other network properties might be of interest for software evolution analysis, which are currently not provided by SEANets. However, given that the Eclipse plug-in contains a full representation of the corresponding graph any other kind of graph metric can be implemented easily. Currently the results repository is public allowing access to projects that have been analyzed by others. Authenticated access to the database could ensure that each researcher views only the data that he/she has provided. Moreover, the current approach according to which the user has to download and import individually all versions that he wishes to analyze, limits scalability. SEANets could benefit from an automated approach where versions would be automatically extracted from a software repository.

A future line of research is the incorporation of software growth models by sampling the distributions of past versions in order to predict future software evolution in terms of network properties. The enhancement of these models by adding domain knowledge for object-oriented systems could also be customizable in terms of rules that can be enabled to investigate their impact on the accuracy of the models.

VII. Conclusion

In this paper we have presented SEANets, an Eclipse plugin that allows software evolution analysis in terms of network properties. The results are available through any Web browser in the form of tables, charts and network visualizations. The generated reports provide insight into evolutionary trends and phenomena governing software growth.

The tool has been tested on several open source projects of different size in terms of the number of generated nodes and edges. Initial evidence indicates that the obtained results offer added value to the already existing techniques for software evolution analysis and the tool performs in a reliable and efficient manner even for large systems.

Tool Website

The SEANets Eclipse plugin and the accompanying video screencast can be downloaded from http://java.uom.gr/seanets/. The page contains also installation instructions and acts as a gateway to the results of already analyzed projects.

References

[1] M. Anderson, "The Data: Six Billion Friends," IEEE Spectrum, vol. 48, pp. 80, June 2011.

[2] A. Chatzigeorgiou, N. Tsantalis, and G. Stephanides, "Application of Graph Theory to OO Software Engineering," Proc. IEEE Work. Interdisciplinary Software Engineering Research (WISER'06), May 2006, pp. 29-35.

[3] A. Chatzigeorgiou and G. Melas, "Trends in Object-Oriented Software Evolution: Investigating Network Properties", Proc. 34th IEEE Int. Conf. on Software Engineering (ICSE 2012), NIER Track, June 2012.

[4] P. Louridas, D. Spinellis, and V. Vlachos, "Power laws in software," ACM T Softw Eng Meth, vol. 18, pp. 1-26, September 2008.

[5] D. Easley and J. Kleinberg, Networks, Crowds, and Markets: Reasoning about a Highly Connected World, Cambridge University Press, 2010.

[6] Ucinet Software, June 2012, https://sites.google.com/site/ucinetsoftware

[7] Gephi, an open source graph visualization and manipulation software, 22 June 2012, http://gephi.org/

[8] Networks/Pajek, June 2012, http://vlado.fmf.uni-lj.si/pub/networks/pajek/

[9] GUESS: The Graph Exploration System, June 2012, http://graphexploration.cond.org/

[10] SNAP: Stanford Network Analysis Project, June 2012, http://snap.stanford.edu/

[11] NetworkX 1.6 documentation, June 2012, http://networkx.lanl.gov/

[12] NetMiner - Premier Software for Social Network Analysis, June 2012, http://www.netminer.com/index.php

[13] The igraph library for complex network research, June 2012, http://igraph.sourceforge.net/

[14] Cytoscape: An Open Source Platform for Complex Network Analysis and Visualization, 22 June 2012, http://www.cytoscape.org/

[15] A. González-Torres, R. Therón, M. Wermelinger and Y. Yu, "Maleku: An Evolutionary Visual Software Analytics Tool for Providing Insights into Software Evolution", Proc. IEEE Int. Conf. on Software Maintenance (ICSM'11), September 2011, pp. 594-597.

[16] M. Pinzger, H. Gall, M. Fischer, and M. Lanza, "Visualizing multiple evolution metrics," Proc ACM Symp. on Software visualization (SoftVis 05), May 2005, pp. 67–75.

[17] K. Lieberherr and I. Holland, "Assuring good style for object-oriented programs," IEEE Software, vol. 6, pp.38–48, September 1989.

[18] FreeCol, 25 June 2012, http://www.freecol.org

[19] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations," Proc. ACM Int. Conf. Knowledge Discovery in Data Mining (KDD 05), August 2005, pp. 177-187.

[20] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," Science, vol. 286, pp. 509-512, October 1999.

[21] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins, "Microscopic Evolution of Social Networks," Proc. ACM Int. Conf. Knowledge Discovery in Data Mining (KDD 08), August 2008, pp. 462-470.

[22] J. Kleinberg, "The Small-World Phenomenon: An Algorithmic Perspective," Proc. ACM Symposium on Theory of Computing (STOC 00), May 2000, pp. 163–170.

[23] InfoVis, JavaScript InfoVis Toolkit, 22 June 2012, http://thejit.org/

[24] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement", Software: Practice and Experience, vol. 21, November 2001, pp. 1129-1164.