Softwarenaut: Exploring Hierarchical System Decompositions

Mircea Lungu, Michele Lanza
Faculty of Informatics, University of Lugano, Switzerland
mircea.lungu@lu.unisi.ch, michele.lanza@unisi.ch

1. Introduction

Softwarenaut is a tool used for the top-down exploration of large software systems. Using it, the reverse engineer can obtain various architectural views of a system by interactively navigating a hierarchical decomposition of the system. In order to accommodate many possible decompositions of a system Softwarenaut was designed as a framework which provides visualization and exploration services that can be applied on various hierarchical decompositions of the system.

2. Exploration mechanisms

There are multiple hierarchical decompositions of a system and for most of them the exploration mechanisms do not depend on the decomposition type. From the mechanisms that Softwarenaut employs we mention the exploration primitives, the multiple perspectives and the visual annotations.

Multiple perspectives. Softwarenaut provides three coupled, complementary perspectives on the analyzed system. As Figure 1 shows, the perspectives are:

- **Exploration Perspective.** Presents a graph-like representation of the visible modules (nodes in the graph) and their interaction (edges in the graph).
- **Detail perspective.** Offers detail on demand for the current selected entity by providing alternative views on it.
- **Map Perspective.** Presents the positions in the hierarchy of the visible modules in order to offer a sense of context and orientation.

Navigation Primitives. There are three principal operations that the framework provides during the navigation.

- **Expand.** Through expansion, a node is replaced with nodes representing its children.
- **Collapse.** By collapsing a node corresponding to a package, the node, together with all the nodes representing the siblings of the package are removed from the view and replaced with a node representing the parent package.
- **Filter.** Because filtering is an important part of exploration, Softwarenaut implements multiple types of filters.

Visual Queries for Annotating Navigation. For the elements in the view, the user can define queries which have as a result, the visual annotation of the conforming elements. The top-left corner of Figure 1 presents the mapping between the queries and the color of the auras around the elements.

3. Types of Analysis

Depending on the available data, on the desired depth of the analysis and on the purpose of the reverse engineering process, various types of analysis are possible. In this section we briefly present several types of analysis that Softwarenaut can perform.

- **Package Dependency.** When analyzing a Java system, the modularization units with the highest gran-
Figure 2. Exploration Perspective in one industrial case study.

Figure 1 presents Softwarenaut exploring the package hierarchy of Azureus, an open source file sharing system of approximately 2500 classes.

**Directory Include Relationships.** For C/C++ projects an initial overview of the system can be obtained by considering the include relationships between files aggregated at directory level. This type of analysis is important for analyzing very large industrial systems because the data can be obtained by a lightweight parsing of the system.

Figure 2 is a snapshot of the Exploration View during the analysis of a large industrial C/C++ system (more than 2MLOC). The nodes represent modules (directories) in the system, and the edges represent include relations aggregated from the contained files. The area of the nodes is proportional to the corresponding module’s size and the width of the edge is proportional to the number of dependencies abstracted in it.

With these conventions in place, the figure shows that, besides having similar size, the two bottom modules also interact in a similar way with the other modules in the system. A reverse engineer might want to understand the causes of such a similarity.

**Semantic Cluster Interaction.** Latent Semantic Indexing (LSI) is an information retrieval technique used primarily in web search. By using LSI to compute the similarity between documents, Kuhn et. al [1] create a hierarchical clustering of the classes in the system on a semantic base. Their aim is to identify concepts in the system.

Because the concept detection process cannot be automated, the solution is to manually inspect the cluster hierarchy. In our previous work [2] we show exploration with Softwarenaut is useful in detecting concepts in a semiautomatic way in a system.

4. Presentation Description

During the presentation, we will interactively reverse engineer a software system using Softwarenaut. The system will be a large (> 1000 classes) software system. We believe that doing the analysis in an interactive way and involving the participants will provide us with important feedback on the strong and weak points of the tool and the methodology.

**Note:** Softwarenaut is open source software, written in Smalltalk. The tool is available for all the major platforms (Windows, Linux, OS X) and can be freely downloaded from: http://www.inf.unisi.ch/phd/lungu/softwarenaut

References
