TopicViewer: Evaluating Remodularizations Using Semantic Clustering

Gustavo Jansen de S. Santos[1], Katyusco de F. Santos[2], Marco Tulio Valente[1], Dalton D. S. Guerrero[3], Nicolas Anquetil[4]

[1]Federal University of Minas Gerais

[2]Federal Institute of Pernambuco

[3]Federal University of Campina Grande

[4]RMoD Team, INRIA

gustavojss@dcc.ufmg.br, katyusco@recife.ifpe.edu.br, mtov@dcc.ufmg.br,

dalton@dsc.ufcg.edu.br, nicolas.anquetil@inria.fr

Abstract. *Software visualization techniques have been proposed to improve program comprehension, as large systems get difficult to understand and maintain. In this paper, we describe the design and main features of* `TopicViewer`*, a tool that uses Information Retrieval techniques and Hierarchical Clustering to show how domain concepts are disposed across one system's architecture as it evolves. Also, we describe an application of this tool during the remodularization of* `JHotDraw`*.*

1. Introduction

As a software system gets larger and more complex, required modifications and improvements get more difficult to implement. It is estimated that up to 60% of software maintenance efforts is spent on understanding the code to be modified [Abran *et al.* 2001] . Most of this knowledge is often spread in external documentation, like task reports.

Many approaches have been proposed to make program comprehension more effective. In general, the main goal is to extract a mental, high-level model of the software, based on documents or by the reverse engineering the source code. On the other hand, to help the interpretation of many documents described in natural language, Information Retrieval (IR) techniques can be used as support to search and acquire similar documents, given a set of information queries.

In order to assist program comprehension tasks, this paper presents `TopicViewer`, a tool that supports a methodology proposed by Kuhn *et al.* [2007] . It relies on IR and Hierarchical Clustering to extract groups of similar documents and retrieve a set of frequent words from these groups, which represent domain concepts in the software. We performed an adaptation of this methodology to make possible its application in any sources of textual information, like bug reports or high-level documentation, and also to support the analysis of how domain concepts evolve after a remodularization.

The remainder of this paper is structured as follows. In Section 2 we present the design and implementation of the `TopicViewer` tool, including the original methodology and our experience with `JHotDraw`'s source code. Section 3 discusses related tools and Section 4 presents our final remarks.

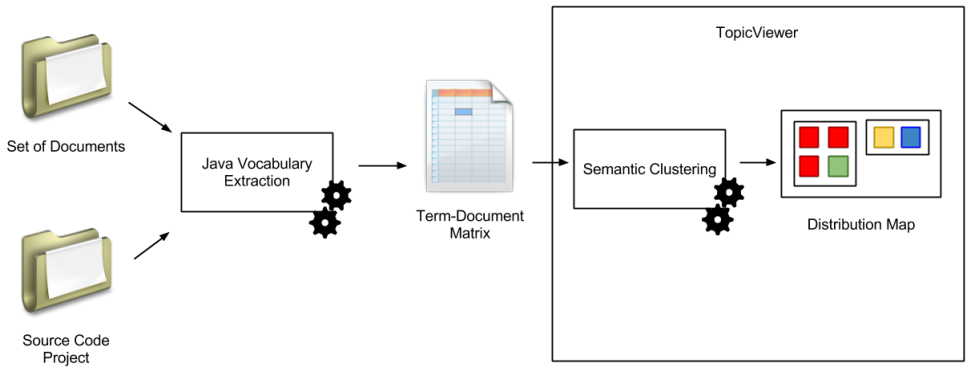


Figure 1. The `TopicViewer` engine

2. The `TopicViewer` Tool

`TopicViewer` is a tool that provide visualizations in terms of clusters of classes that represent common semantic concerns. Basically, classes that manipulate the same concerns (e.g. data structures, file management) are tipically grouped in the same cluster.

For this purpose, `TopicViewer` requires a representation of the set of documents as a term-document matrix, which each cell `(t,d)` contains the relevance of term `t` in the document `d`. In this work, a software entity is a container of inner entities or identifiers, like packages, classes, files or methods. Moreover, although we focus on source code text extraction, our approach can be applied to bug reports and other textual documents. This matrix must be preprocessed, as the text extraction fully depends on the document's structure. However, an interface of this matrix is provided in order to handle systems with large number of terms and documents.

As presented in Figure 1, the result of `TopicViewer` process is a Distribution Map [Ducasse *et al.* 2006]: a visualization in which documents are organized in folders and packages, and colored according to the group of similar classes to which they belong. The organization of these documents is important in the visualization, in order to analyze the distribution of domain concepts over time.

We present the original methodology in Subsection 2.1. Then, we introduce our adaptations to assess software evolution (Subsection 2.2). Next, we present the design and implementation of `TopicViewer`, and finally a case study using source code extraction and analysis of a `JHotDraw` remodularization (Subsections 2.3 and 2.4).

2.1. Semantic Clustering

Semantic Clustering is a technique proposed by Kuhn *et al.* to group similar classes of a system, according to their vocabularies [2007]. According to the technique, every class is represented as a document, and terms are obtained by extracting and filtering identifiers and comments. Moreover, terms are later weighted with *tf-idf* function, in order to punish words that appear in many documents of the vocabulary. The resulting term-document matrix is then processed by Latent Semantic Indexing (LSI), an IR technique that reduces the matrix in number of terms with minimal loss of information [Baeza-Yates and Ribeiro-Neto 2011].

With the reduced matrix, each class is retrieved as a vector, and the similarity of a pair of classes is calculated by the cosine of the smallest angle formed by their representing vectors. Next, the technique relies on an agglomerative Hierarchical Clustering to group a fixed number of clusters or until no pair of clusters has similarity greater than a given threshold. Finally, queries are used to retrieve a set of words representing the meaning of each group (called semantic cluster) and the Distribution Map is generated.

Ducasse *et al.* also proposed two metrics to measure concentration and spreading of a semantic cluster [2006]. The *Spread* of a cluster is calculated as the number of parts (e.g. folders or packages) covered by the cluster. Moreover, *Focus* measures how a cluster dominates, i.e., covers most of the classes of the parts in which they appear. We also rely on an *Internal Cohesion* metric to assess the quality of a package as the mean similarity between their classes, pairwise [Marcus and Poshyvanyk 2005]. In contrast to structural metrics that consider static dependencies between software artifacts, these metrics express a conceptual relationship between these artifacts, as denoted by common terms in their vocabularies.

2.2. Remodularization Support

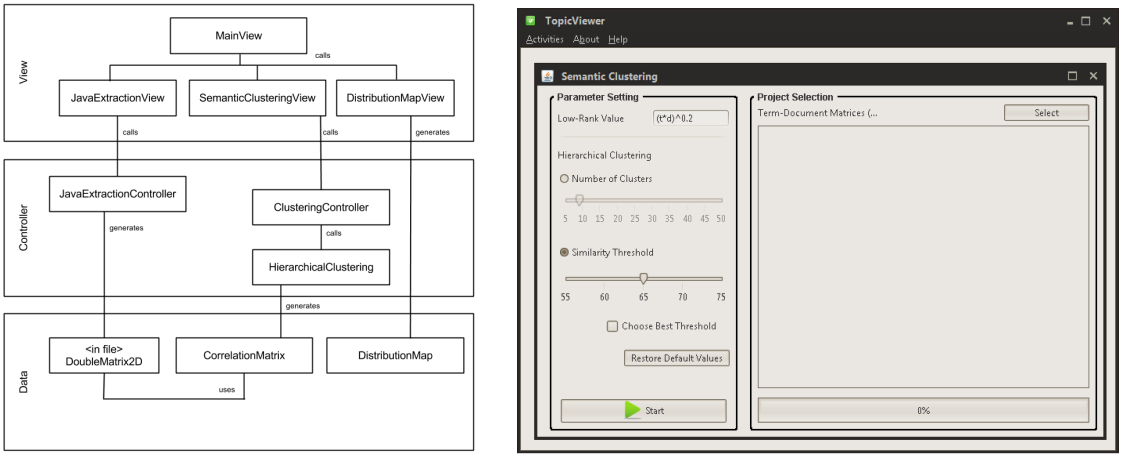
The original Semantic Clustering technique supports the extraction of clusters for a single version of a given system. Since we also want to evaluate how the semantic clusters evolve over time, we adapted this methodology to use two versions of the same system. For this purpose, we calculate the semantic clusters for the first version, and each cluster is represented by a vector that represents the sum of all vectors of this cluster's documents. Then, for the next version, each class vector is mapped to the previous version vocabulary, since their set of terms may differ. Finally, we calculate the cosine similarity between this mapped class and all cluster vectors from the first version, and we assign the class to the cluster with the highest similarity.

The result is a representation of two Distribution Maps. New packages or classes are displayed in both versions, but as empty entities in the first one. Similarly, removed packages or classes are displayed as empty entities in the second version. It is important to observe how the disposition of domain concepts evolves between versions. In other words, we expect that the classes of a package are more internally similar after maintenance tasks, and so the concentration of domain concepts.

2.3. Internal Architecture and Interface

`TopicViewer` is a desktop application implemented in Java. To use the tool, developers must follow four steps: (i) *Set Workspace*, i.e., a folder in which all output data is stored; (ii) *Java Vocabulary Extraction*, to extract vocabulary from Java source code, (iii) *Semantic Clustering*, in order to generate semantic clusters and Distribution Map given a term-document matrix, and (iv) *Distribution Map Viewer*, to interact with the map structure described in Subsection 2.1.

Our tool reuses IR functions from `VocabularyTools` [Santos *et al.* 2012], a set of tools for extraction, filtering and LSI operations in Java, developed by the Software Practices Lab of Federal University of Campina Grande (UFCG). The current `TopicViewer` implementation follows a Model-View-Controller architectural pattern with three main modules, as presented in Figure 2a:



(a) Architecture (b) User Interface

Figure 2. `TopicViewer` Architecture and User Interface

- *Data*: Representation of the main data structures of the tool. It includes matrix storage and operations in file, representation of a Distribution Map and utility functions to save these data structures and other information in files.
- *Controller* contains the main algorithms: the agglomerative Hierarchical Clustering, semantic topics extraction, as well as an interface for `VocabularyTools` operations.
- *View*: A set of Swing windows providing an user interface.

The interface for Semantic Clustering is shown in Figure 2b. The developer can choose (in the left panel) which clustering stop criteria to use, as a fixed number of clusters or by a similarity threshold. Next, the developer can choose a set of term-document matrices to cluster (right panel) and monitor the execution progress on the bottom bar.

2.4. Application

In this section we detail our experience with `TopicViewer` in evaluating a remodularization of `JHotDraw`, a framework for technical drawing editors[1]. We focused on the remodularization from version 7.3.1 to 7.4.1 in this paper. We used the `TopicViewer` tool to extract the semantic clusters and to generate the comparative Distribution Map, as displayed in Figure 3.

We observed that a lot of packages were created in this remodularization, as they are empty in the first map. Most of them have their internal classes very similar, and the classes are mapped to a common cluster. To assist our analysis, we provided an interaction with *Distribution Map View* so the user can visualize quality metric values for both versions (see Section 2.1), as comparison means to assess maintenance quality.

In this remodularization, the packages *org.jhotdraw.app.action* and *org.jhotdraw.draw* have been split up into sub-packages, resulting in the creation of 16 packages. For example, the *app.action* package had internal cohesion of 0.89 in the 7.3.1 version; after the remodularization, it had 0.94 and the average cohesion of the

[1] http://www.jhotdraw.org/

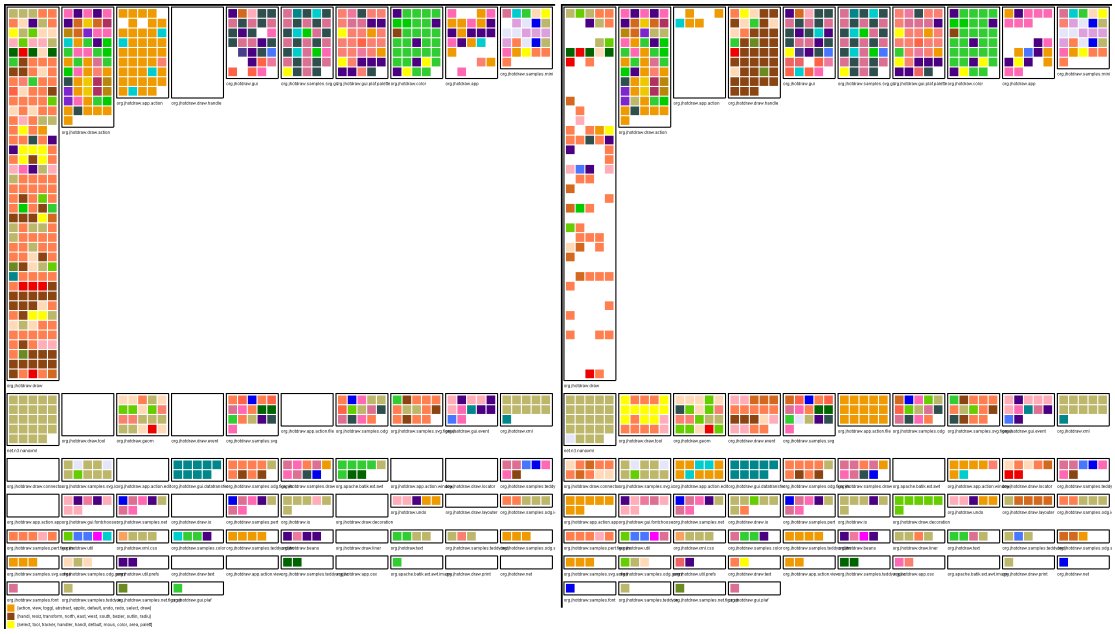


Figure 3. Comparative Distribution Map for `JHotDraw-7.3.1 - 7.4.1` Remodularization. A colored version of this image with better resolution is available at http://tinyurl.com/c4efph8

splitted packages is 0,98. It means that, internally, their classes are more cohesive in their respective new packages. Moreover, analyzing the semantic topic that dominated that package (marked in orange), both metrics Spread and Focus increased, which means that this cluster covers a greater number of packages, and the concern attached to it dominates the new packages after the remodularization effort.

As the latter refactored package, *draw*, its internal cohesion decreased a from 0.58 to 0.57. Nevertheless, the average cohesion of the eleven created packages from this refactoring is 0,70, which means that their classes are more cohesive. For the semantic clusters, both values of Spread and Focus increased.

For the semantic clusters, the same happens for the most dominating clusters, marked in yellow and dark chocolate, in which both Spread and Focus values increased. In general, we observed that semantic clusters usually share the same behavior in this case study: they are spread in a higher number of packages, and also dominate more classes in each package, increasing their Focus.

3. Related Tools

`TopicXP` is an Eclipse plugin that uses Latent Dirichlet Allocation and term extraction from identifiers and comments in source code [Savage *et al.* 2010]. The semantic topics can be shown in different views in which the user can inspect the relationship between them and excerpts in the code. `CodeTopics` is also an Eclipse plugin that uses information from high-level artifacts, i.e., textual documents provided by developers, and shows a mapping between these artifacts and the source code [Gethers *et al.* 2011].

At some extent, `Hapax` is the most similar tool to `TopicViewer`, and it was developed as a plugin for the `Moose` platform [Kuhn *et al.* 2007]. It performs text extrac-

tion from identifiers and comments, and returns a Distribution Map as result. Nonetheless, `TopicViewer` provides a comparative view to assist software evolution assessment, under a semantic point of view.

4. Final Remarks

Program comprehension often requires retrieving a set of external documents, most of them in natural language. Usually, it is difficult to match the semantics of these documents and the intention (or meaning) of the source code. Various approaches combine Information Retrieval and Software Visualization techniques to overcome these difficulties, as a means to assist the understanding and assess the conceptual quality of a code.

We developed `TopicViewer`, a tool that uses Latent Semantic Indexing to extract groups of similar classes, according to their vocabularies. We applied our tool to the remodularization of a real system, and we could highlight the changes, as well as assess this remodularization in terms of concept concentration and conceptual internal quality of the refactored packages.

`TopicViewer` is publicly available at http://code.google.com/p/topic-viewer/.

Acknowledgments:

Our research is supported by CAPES, FAPEMIG, and CNPq.

References

Abran, A., Bourque, P., Dupuis, R., and Moore, J. W., editors (2001). *Guide to the Software Engineering Body of Knowledge - SWEBOK*. IEEE Press, Piscataway, NJ, USA.

Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (2011). *Modern Information Retrieval - The concepts and technology behind search, Second edition*. Pearson Education Ltd., Harlow, England.

Ducasse, S., Gîrba, T., and Kuhn, A. (2006). Distribution map. In *22nd IEEE International Conference on Software Maintenance*, pages 203–212.

Gethers, M., Savage, T., Di Penta, M., Oliveto, R., Poshyvanyk, D., and De Lucia, A. (2011). Codetopics: which topic am I coding now? In *33rd International Conference on Software Engineering*, pages 1034–1036.

Kuhn, A., Ducasse, S., and Gîrba, T. (2007). Semantic clustering: Identifying topics in source code. *Information & Software Technology*, 49(3):230–243.

Marcus, A. and Poshyvanyk, D. (2005). The conceptual cohesion of classes. In *21st IEEE International Conference on Software Maintenance*, pages 133–142.

Santos, K. D. F., Guerrero, D. D. S., Figueiredo, J., and Bittencourt, R. A. (2012). Towards a prediction model for source code vocabulary. In *1st International Workshop on the Next Five Years of Text Analysis in Software Maintenance, International Conference on Software Maintenance*.

Savage, T., Dit, B., Gethers, M., and Poshyvanyk, D. (2010). TopicXP: Exploring topics in source code using latent dirichlet allocation. In *26th IEEE International Conference on Software Maintenance*, pages 1–6.