# CodeTopics: Which Topic am I Coding Now?

Malcom Gethers
Computer Science Dept.
College of William and Mary
Williamsburg, VA, USA
mgethers@cs.wm.edu

Trevor Savage
HCI Institute
Carnegie Mellon University
Pittsburgh, PA, USA
trevorsa@cs.cmu.edu

Massimiliano Di Penta
Dept. of Engineering
University of Sannio
Benevento, Italy
dipenta@unisannio.it

Rocco Oliveto
STAT Dept.
University of Molise
Pesche (IS), Italy
rocco.oliveto@unimol.it

Denys Poshyvanyk
Computer Science Dept.
College of William and Mary
Williamsburg, VA, USA
denys@cs.wm.edu

Andrea De Lucia
Software Engineering Lab
University of Salerno
Fisciano (SA), Italy
adelucia@unisa.it

## ABSTRACT

Recent studies indicated that showing the similarity between the source code being developed and related high-level artifacts (HLAs), such as requirements, helps developers improve the quality of source code identifiers. In this paper, we present CodeTopics, an Eclipse plug-in that in addition to showing the similarity between source code and HLAs also highlights to what extent the code under development covers topics described in HLAs. Such views complement information derived by showing only the similarity between source code and HLAs helping (i) developers to identify functionality that are not implemented yet or (ii) newcomers to comprehend source code artifacts by showing them the topics that these artifacts relate to.

## Categories and Subject Descriptors

D.2.3 [**Software Engineering**]: Coding Tools and Techniques—*Program editors*

## General Terms

Documentation; Measurement; Management.

## Keywords

Source code lexicon, program comprehension, traceability.

## 1. INTRODUCTION

Consistent use of identifiers and detailed, meaningful comments are two factors that can affect source code maintainability and comprehensibility, as indicated by recent studies [6, 9]. For example, referring to a concept using a non-meaningful term, or either using different terms to refer to the same concept, may increase the program comprehension burden. This also creates a mismatch between the developers' cognitive model and the intended meaning of the term, thus increasing the risk of fault proneness [9].


*ICSE* '11, May 21–28, 2011, Waikiki, Honolulu, HI, USA
ACM 978-1-4503-0445-0/11/05.

Recent studies indicated that showing the similarity between the source code being developed and related high-level artifacts (HLAs) helps developers improve the quality of source code identifiers [3]. In particular, De Lucia *et al.* have proposed COCONUT [3], an Eclipse plug-in that continuously shows the similarity between the source code that the developer is writing and related HLAs (e.g., use cases, requirements, design documents). This similarity information might induce the developer to take different actions, such as making the source code identifiers more consistent with domain terms or better commenting the source code. However, such a similarity information might be too coarse-grained: a source code artifact the developer is coding may by related to only one specific concept described in one HLA (and not to the whole HLA), or, *vice versa*, it may be related to concepts spread across different HLAs.

We conjecture that a better support can be provided showing not only the similarity between source code and HLAs, but, going into a finer-grained level of detail, showing what are the specific topics being covered by the source code, and to what extent these topics are covered. In this demo, we present CodeTopics—an Eclipse plug-in built on top of COCONUT—that uses an Information Retrieval (IR) technique, namely Relational Topics Model (RTM) [2]—to analyze the topics that software artifacts (i.e., HLAs and source code) deal with. Recently, RTM has been used to capture coupling in object-oriented systems [5], whereas topic models have been also used to capture cohesion in object-oriented systems [7], explore topics in source code [8], and support recovery of traceability links between software artifacts [1].

CodeTopics—other than showing the similarity between source code and related HLAs as COCONUT does—visualizes the topics found in HLAs—by means of a list of terms describing each topic—and highlights how important a topic is for a HLA and to what extent the topic has been covered in the source code artifact under development. This can be used by developers to address the following needs:

- for a new developer joining the project, it describes the source code file that the developer has opened in the IDE using terms from the HLA traced to the source code, i.e., provides an automatic, high-level description of the source code, thus aiding the comprehension;

- it shows how the source code being developed covers specific *concepts* described in a requirement/use cases. This information complements the similarity between source code and HLAs, giving a finer-grained level of details that might improve the support provided to the developer to improve the source code identifiers [3].

- it helps to check whether all the concepts described in HLAs have source code related to them. In case where there are concepts for which no correspondence in the source code exists, this information can be used to identify high-level, abstract concepts of features that have not been implemented yet. On the other way around, it also shows if there are concepts (implementation details) only mapped to source code and not to HLAs.

Summarizing, the specific contributions of this tool demo paper are: (i) a technique for analyzing and understanding high-level topics and their relationships, as extracted with RTM, that can be used for improving program comprehension and/or for detecting concepts and features that are not implemented in source code, but which perhaps should be; and (ii) the Eclipse plug-in, namely CodeTopics, which implements an RTM-based set of features for evaluating lexicon used in source files and HLAs.

## 2. CODETOPICS IN A NUTSHELL

This section describes CodeTopics. The current implementation of the tool supports Java development (as it is integrated in the Eclipse Java Development Kit) although without loss of generality the proposed approach can be extended to other programming languages. The following subsections explain how the tool extracts the needed information from the source code and from HLAs (Section 2.1), and then describes the views provided by CodeTopics (Section 2.2).

### 2.1 Topic Extraction

Our underlying mechanism for establishing topics, namely RTM, accepts as input a collection of documents (i.e., a corpus) and a set of traceability links between documents. The documents utilized in CodeTopics include HLAs (e.g., use cases) as well as all the Java source code files associated with the project currently opened in Eclipse-JDT.

To identify similarities between HLA and source code, and to extract topics, RTM needs pre-processed artifacts, plus a list of links between artifacts. Source code files are processed to filter out everything, but identifiers and comment words, while all HLA are filtered using a list of stop-words.

Traceability links are obtained as follows. HLA-to-source code links, are provided by the user. Links between HLA are automatically recovered using a technique based on the similarity between artifacts computed using Latent Semantic Indexing (LSI) [4]. Links between source code classes are method call dependencies recovered using X-Ray[1]. Finally, in addition to the automatic link identification, we allow users to specify additional links or to remove automatically recovered ones.

[1]http://xray.inf.usi.ch/xray.php

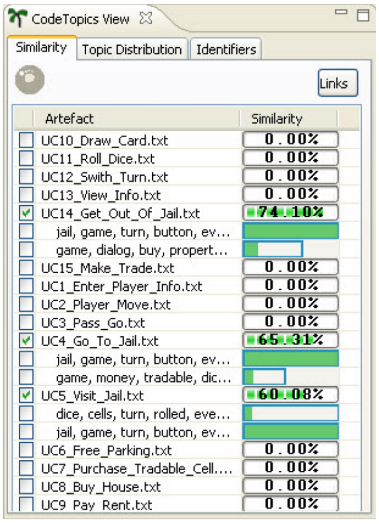


**Figure 1: The Similarity view.**

For the generation of the RTM model, we use the R package *lda*[2], which implements several information retrieval algorithms, including RTM.

### 2.2 CodeTopics at Work

To support the developer in writing source code, and specifically to support her specific programming needs outlined in the introduction, CodeTopics provides developers with two different views: *Similarity* and *Topic Distribution* views. In addition, CodeTopics inherits from COCONUT the *Identifiers* view, which recommends a list of candidate identifiers extracting $n$-grams from HLAs related to the source code under development. In the following we describe the *Similarity* and *Topic Distribution* views using use cases and Java source code from a simple program, that is Monopoly game[3]. A more detailed description of CodeTopics at work is provided in a video available online[4].

#### 2.2.1 Similarity View

The aim of the *Similarity* view (Figure 1) is to show (i) as in COCONUT, how similar is the code under development to the related HLAs, and (ii) how the code under development relates to the topics found in the HLAs it is traced to, thus providing a quick description of the code in terms of words from the related requirements. This view is context-sensitive — its content relates to the source file active in the Eclipse-JDT editor.

The view shows a list of high-level artifacts, with a checkbox—in the first column—indicating whether the artifact is related to the source code under development. The second column contains the name of the HLAs, and the third column shows the similarity between the artifact and the source code under development (or zero if the HLA is not linked to the source code). The indented rows beneath each selected high-level artifact describe—by using a list of terms— the topics found in the HLA. For each topic, the *Similarity* view shows the degree to which the topic is found in the design ar-

[2]http://cran.r-project.org/web/packages/lda/
[3]http://agile.csc.ncsu.edu/rose/#reales
[4]http://www.youtube.com/watch?v=guU8Atqo7xY

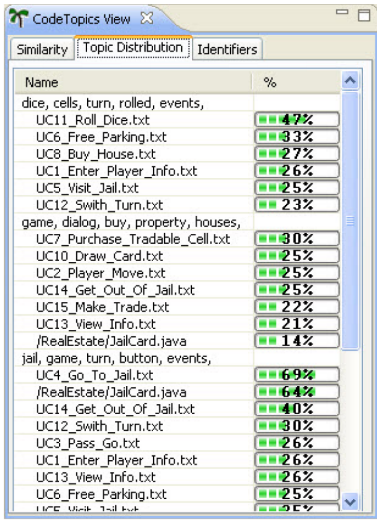


**Figure 2: The Topic Distribution view.**

tifact (length of the blue rectangle) and in the source code file (length of the green bar, which at its greatest length would fill blue rectangle). This would provide additional information to the developer, other than the similarity between code and requirements: given a HLA and a source code file under development, the view provides an indication of the importance of different topics in the high-level artifact, and the degree to which the code relates to those topics.

The example of Figure 1 shows that the class under development (*JailCard*, which models the jail in the monopoly game) has, indeed a high similarity with use cases related to managing the "going to jail" event in the game (*UC4*, *UC5*, *UC14*). These use cases contain 2 topics, one, more important (according to the length of the blue rectangles)—described by words *jail*, *game*, *turn*, *event* related to going to jail event, and the other related to more general parts of the monopoly game, like rolling the dice, paying/earning money, buying/renting lands/houses, etc. In this case, we can see that the class *JailCard* covers completely the first topic, while it is less related to the second one.

### 2.2.2 Topic Distribution View

The *Topic Distribution* view (Figure 2) provides information from an opposite viewpoint as compared to the *Similarity* view. In particular, this view shows a table reporting the topics identified by RTM in all the artifacts (both documentation and source code) and, for each identified topic, the degree to which the topic is found both in HLAs and in source code artifacts. This shows which are the artifacts more related to a topic (by means of bars). In addition, it indicates whether a topic is not reflected in the source code (i.e., there is no source code below a topic), as in the case of the *dialog, lands, after, amount, ...* topic in our example, either because this is a high level concept, or because the concept has not been implemented yet. Similarly, it can indicate cases where the topic was only found in the code—i.e., no high-level artifact mentioned below the topic—and not in the HLA. This can happen for topics related to implementation details such as algorithms, protocols, etc.

# 3. CONCLUSION

In this paper we presented CodeTopics, a tool which supplies programmers with topics found in the source code under development, using the Relational Topics Model technique. The information provided by CodeTopics has different purposes, namely (i) helping newcomers to quickly get an idea of what a source code artifact is about, by describing it in terms of topics found in related high-level artifacts, e.g., requirements or use cases; (ii) providing a finer-grained level of traceability, i.e., telling to what extent a source code artifact is related to a topic, other than to what extent it is related to a high-level artifact; and (iii) highlighting topics not covered by source code (high-level concepts) as well as topics only covered by source code (implementation details). CodeTopics is publicly available for download[5].

Future work aims at further improving CodeTopics with more advanced views, e.g., showing how well requirements are (overall) covered by source code. Also, we plan to conduct controlled experiment to evaluate the the usefulness of CodeTopics tool in program comprehension and development tasks.

# 4. ACKNOWLEDGEMENTS

Chappell Fellowship program of the College of William and Mary provided funding for Trevor Savage working on his undergraduate research project. This work was supported in part by NSF CCF-1016868 grant. Any opinions, findings, and conclusions expressed herein are the authors' and do not necessarily reflect those of the sponsors.

# 5. REFERENCES

[1] H. U. Asuncion, A. Asuncion, and R. N. Taylor. Software traceability with topic modeling. In *ICSE'10*, pages 95–104, 2010.

[2] J. Chang and D. M. Blei. Hierarchical relational models for document networks. *Annals of Applied Statistics*, 2010.

[3] A. De Lucia, M. Di Penta, and R. Oliveto. Improving source code lexicon via traceability and information retrieval. *TSE*, (to appear), 2011.

[4] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Recovering traceability links in software artefact management systems using information retrieval methods. *TOSEM*, 16(4), 2007.

[5] M. Gethers and D. Poshyvanyk. Using relational topic models to capture coupling among classes in object-oriented software systems. In *ICSM'10*, 2010.

[6] D. Lawrie, H. Feild, and D. Binkley. Quantifying identifier quality: an analysis of trends. *ESE Journal*, 12(4):359–388, 2007.

[7] Y. Liu, D. Poshyvanyk, R. Ferenc, T. Gyimóthy, and N. Chrisochoides. Modeling class cohesion as mixtures of latent topics. In *ICSM'09*, pages 233–242, 2009.

[8] T. Savage, B. Dit, M. Gethers, and D. Poshyvanyk. Topicxp: Exploring topics in source code using latent dirichlet allocation. In *ICSM'10*, 2010.

[9] A. Takang, P. Grubb, and R. Macredie. The effects of comments and identifier names on program comprehensibility: an experiential study. *Journal of Program Languages*, 4(3):143–167, 1996.

[5] http://www.cs.wm.edu/semeru/CodeTopics