Adding Examples into Java Documents

Jinhan Kim, Sanghoon Lee, Seung-won Hwang
Pohang University of Science and Technology
Pohang, Korea
{*wlsgks08,sanghoon,swhwang*}*@postech.edu*

Sunghun Kim
Hong Kong University of Science and Technology
Hong Kong
hunkim@cse.ust.hk

***Abstract*—Code examples play an important role to explain the usage of Application Programming Interfaces (APIs), but most API documents do not provide sufficient code examples. For example, for the JDK 5 documents (JavaDocs), only 2% of APIs have code examples. In this paper, we propose a technique that automatically augments API documents with code examples. Our approach finds and embeds code examples for more than 75% of the APIs in JavaDocs 5.**

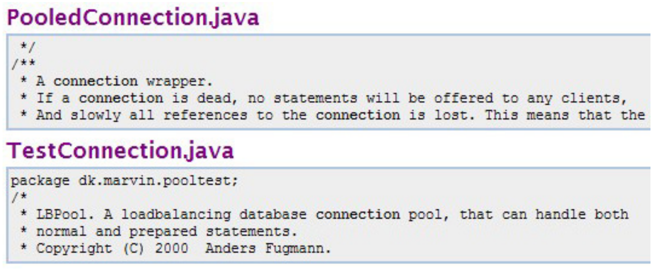
***Keywords*-API Documents; Examples; Structures; Code Clustering; Ranking;**

I. Introduction

Developers often reuse existing libraries such as Application Programming Interfaces (APIs) to improve the productivity and quality of code [11], [10]. However, it is difficult for developers to use APIs correctly without understanding the intention and usage of APIs. For this reason, most APIs provide documents, which describe input arguments, output values, and the goal of the APIs. However, some API documents are ambiguous, which may lead to misuse, because most of them are written in a human readable language. As a result, developers usually seek additional formal information, such as code examples.

To satisfy these needs, MSDN from Microsoft [6] and Leopard Reference Library from Apple [5] include a rich set of code examples written by experts. Since making proper code examples for all APIs in documents requires significant human effort, most API documents often do not provide enough code examples. For example, JDK 5 documents (JavaDocs) contain descriptions of more than 27,000 APIs, but only 2% of them (around 500 API descriptions) include code examples.

As a result, developers seek additional code examples using code search engines such as Koders [7] and Google Code Search [1]. However, it is difficult for developers to find appropriate code examples. For example, suppose a developer want to find code examples of API "prepareStatement()" at the interface "Connection" and send a query "Connection prepareStatement". Figure 1 shows the top two results of Koders [7]. The snippets of the top two results highlight the comments in the code and do not provide any usage information of "prepareStatement()". For this reason, developers typically need to send several queries and


PooledConnection.java

```
 */
/**
 * A connection wrapper.
 * If a connection is dead, no statements will be offered to any clients,
 * And slowly all references to the connection is lost. This means that the
```

TestConnection.java

```
package dk.marvin.pooltest;
/*
 * LBPool. A loadbalancing database connection pool, that can handle both
 * normal and prepared statements.
 * Copyright (C) 2000  Anders Fugmann.
```

Figure 1. Koders Top-2 results when the query is "Connection prepareStatement"

examine the query results one by one to get proper code examples, which is time comsuming.

To address this problem, automatic code example extraction techniques [13], [21], [22] have been studied. For example, Holmes and Murphy proposed a code example recommendation technique by matching structures of given code [13], and MAPO recommends source code by mining the sequence of APIs [21], [22]. However, these approaches require special tools such as Eclipse plug-in.

In a clear contrast, we aim at augmenting API documents themselves with code examples, which is desirable because of the following two reasons: First, API documents are the first place to get the API usage information. Second, developers do not need any special tool for accessing the augmented API documents.

Toward this goal, in this paper, we propose an automatic technique that extracts suitable code examples from code repositories, and generates *example oriented API documents* (*eXoaDocs*), which contain extracted code examples. We then apply our technique to JDK 5 documents and generate eXoaDocs for JDK 5. The generated eXoaDocs include code examples of more than 20,000 APIs in JDK 5 (75% of entire APIs).

Our contributions can be summarized as follows:

- **Automatic code example generation technique:** We propose a technique that extracts code examples from a repository, extracts semantic features from code examples, clusters them, and finds a representative code example from each cluster.
- **Generated eXoaDocs:** We embed extracted code examples in existing API documents and generate eXoaD-

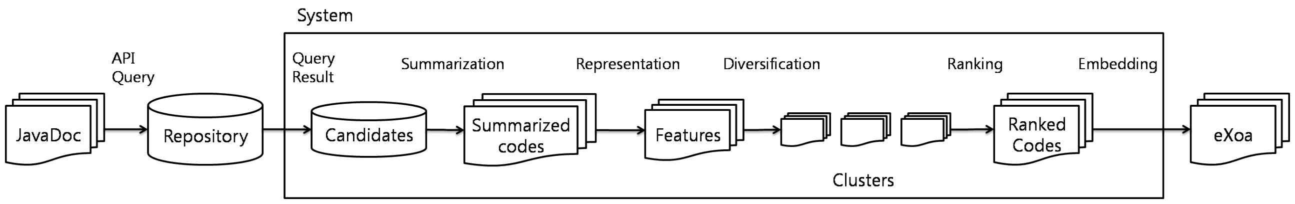


Figure 2. Process Diagram of Automatic Generation of Example API Documents

ocs, augmenting more than 75% of the JDK 5 APIs.

- **Evaluation:** We manually inspect and measure the quantity of generated code examples.

The remainder of this paper is organized as follows. Section II presents the proposed automatic eXoaDocs generation technique. Section III evaluates generated eXoaDocs. Section IV surveys related work, and Section V concludes this paper.

II. Our Approach

This section presents the process of automatic eXoaDocs generation.

Figure 2 shows the overview of our technique that consists of four main modules, summarization, representation, diversification, and ranking. First, we build a repository of candidate code examples by leveraging an existing search engine, Koders, for each API. We then summarize code examples and extract semantic features from code examples. Finally, we cluster and rank code examples to find the most representative code examples from each API.

We describe the details of each module as follows:

- **Summarization:** We summarize candidate code examples into small and relevant snippets. First, from candidate code examples, we identify relevant methods using the given API. If there is no such method, we cannot find code examples for the given API. Second, we analyze the Abstract Syntax Tree (AST) of the identified methods and find related lines to the given API. We slice only these lines and generate summarized code examples for the given API.
- **Representation:** We extract semantic features from the summarized code for clustering and ranking. Semantic features in a code example can be represented in a vector, as explored in DECKARD [17]. Each element in the vector represents the count of occurrence of each specific semantic node type. We exploit these vectors to compare each code to cluster and rank vectors.
- **Diversification:** We cluster the summarized codes into different usage types to present diverse types of usages, using k-means clustering algorithm. However, as it is non-trivial to determine the number of cluster k, we vary k from two to five and choose a desirable number k using heuristic measures.
- **Ranking:** We rank the summarized codes in each cluster to select the most representative code example from each cluster, combining several measures for ranking such as representativeness, conciseness, and correctness.

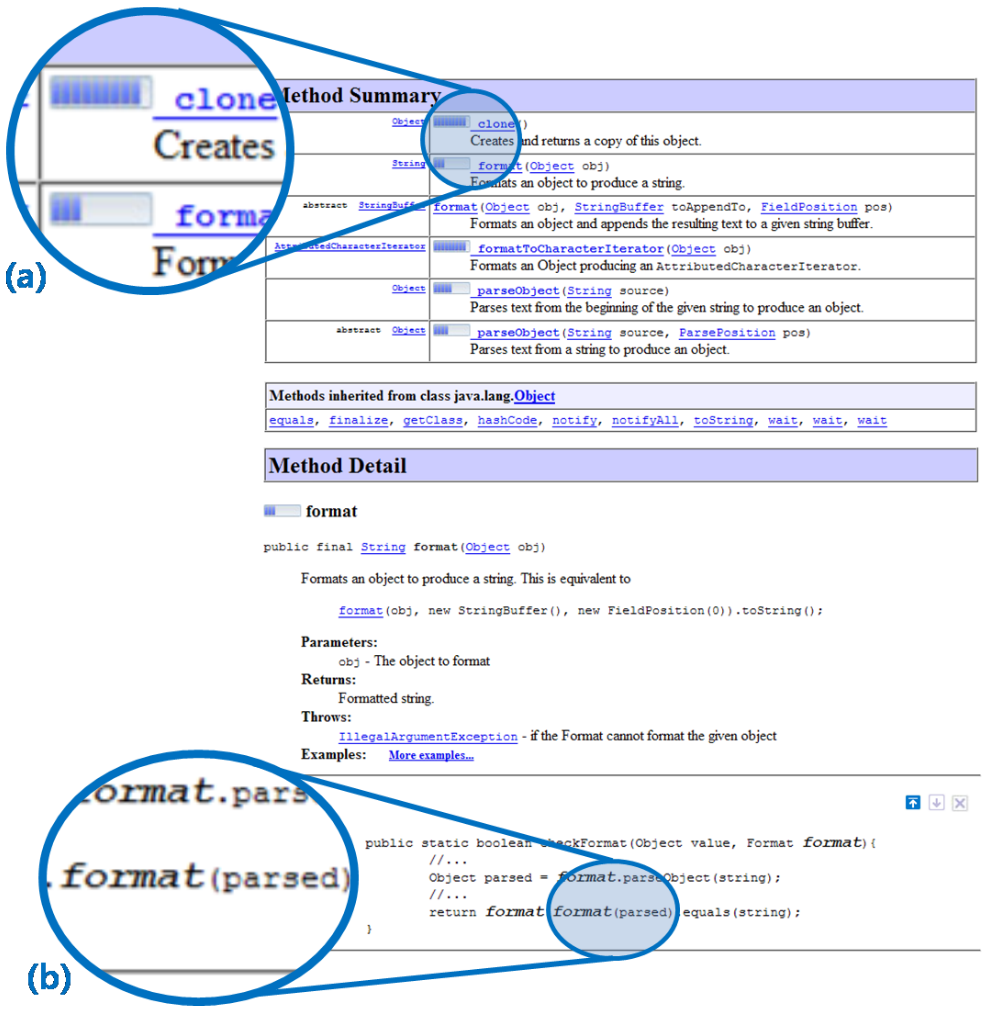
III. Example Evaluation

This section evaluates the quality of code examples in eXoaDocs generated, by manually inspecting the quantity of code examples in eXoaDocs.

Table I
The number of examples in each package in JavaDocs and eXoaDocs

package name	# of methods	# of methods with examples in JavaDocs (%)	# of methods with examples in eXoaDocs (%)
java.applet	45	0 (0.00%)	39 (86.67%)
java.awt	3906	69 (1.77%)	3195 (81.80%)
java.beans	361	6 (1.66%)	335 (92.80%)
java.io	593	20 (3.37%)	571 (96.29%)
java.lang	1133	39 (3.44%)	962 (84.91%)
java.math	112	1 (0.89%)	103 (91.96%)
java.net	452	8 (1.77%)	404 (89.38%)
java.nio	457	16 (3.50%)	405 (88.62%)
java.rmi	467	16 (3.43%)	139 (29.76%)
java.security	790	15 (1.90%)	630 (79.75%)
java.sql	674	7 (1.04%)	590 (87.54%)
java.text	353	12 (3.40%)	305 (86.40%)
java.util	1931	81 (4.19%)	1591 (82.39%)
javax.accessibility	168	0 (0.00%)	122 (72.62%)
javax.activity	0	0 (-)	0 (-)
javax.crypto	215	2 (0.93%)	158 (73.49%)
javax.imageio	722	6 (0.83%)	427 (59.14%)
javax.management	1141	9 (0.79%)	880 (77.13%)
javax.naming	466	22 (4.72%)	321 (68.88%)
javax.net	199	3 (1.51%)	140 (70.35%)
javax.print	432	9 (2.08%)	342 (79.17%)
javax.rmi	73	0 (0.00%)	56 (76.71%)
javax.security	193	12 (6.22%)	122 (63.21%)
javax.sound	418	2 (0.48%)	361 (86.36%)
javax.sql	426	14 (3.29%)	264 (61.97%)
javax.swing	8327	84 (1.01%)	6251 (75.07%)
javax.transaction	13	0 (0.00%)	13 (100.00%)
javax.xml	348	7 (2.01%)	320 (91.95%)
org.ietf	105	4 (3.81%)	60 (57.14%)
org.omg	2356	16 (0.68%)	1029 (43.68%)
org.w3c	265	1 (0.38%)	150 (56.60%)
org.xml	297	9 (3.03%)	195 (65.66%)
total	27438	490 (1.79%)	20480 (74.64%)

Figure 3 illustrates an example page of eXoaDocs. Figure 3(a) shows a popularity bar of each API. Using this bar, developers can easily locate the frequently used APIs. Figure 3(b) shows extracted code examples. About two to five code examples are provided for each API. Developers can find more examples by clicking the *More examples* link.



Method Summary

`Object`	`clone()` Creates and returns a copy of this object.
`String`	`format(Object obj)` Formats an object to produce a string.
`abstract StringBuffer`	`format(Object obj, StringBuffer toAppendTo, FieldPosition pos)` Formats an object and appends the resulting text to a given string buffer.
`AttributedCharacterIterator`	`formatToCharacterIterator(Object obj)` Formats an Object producing an `AttributedCharacterIterator`.
`Object`	`parseObject(String source)` Parses text from the beginning of the given string to produce an object.
`abstract Object`	`parseObject(String source, ParsePosition pos)` Parses text from a string to produce an object.

Methods inherited from class java.lang.Object

`equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Method Detail

format

```
public final String format(Object obj)
```

Formats an object to produce a string. This is equivalent to

```
format(obj, new StringBuffer(), new FieldPosition(0)).toString();
```

Parameters:
`obj` - The object to format
Returns:
Formatted string.
Throws:
`IllegalArgumentException` - if the Format cannot format the given object
Examples: More examples...

```
public static boolean checkFormat(Object value, Format format){
        //...
        Object parsed = format.parseObject(string);
        //...
        return format.format(parsed).equals(string);
}
```

Figure 3. An example page of generated eXoaDocs

We manually inspected the quality of code examples in eXoaDocs for JDK 5. Table I shows the number of code examples in each package. In the original JavaDocs, no package has code examples for more than 5% of the APIs. In contrast, most packages in eXoaDocs have code examples for more than 70% of the APIs and frequently used packages, such as java.io, cover more than 95% of the APIs. Overall, while only 490 APIs in JavaDocs have code examples, more than 20,000 APIs in eXoaDocs have code examples (*i.e.*, 75% of the entire APIs in JavaDocs).

IV. Related Work

A. *Example Recommendation*

MSDN from Microsoft [6], Leopard Reference Library from Apple [5], and PHP API documents [8] provide a large set of code examples written by experts. Alternatively, Java Examples [3] and KodeJava [4] collect code examples from developers. Though these code examples, provided manually by experts, are of high quality and easy to understand, manually providing code examples is time consuming, which explains why, most API documents do not have enough code examples.

There are many automatic code example recommendation systems [13], [19], [21], [22] to address this problem. Holmes and Murphy proposed a source code example recommendation technique by matching structures of given code [13]. XSnippet [19] provides a context-sensitive code assistant framework which recommends sample code snippets. MAPO analyzes source codes and mines frequent API usage sequences [21], [22]. These proposed techniques are similar to ours in the sense that they use their own repositories and recommend code examples automatically.

However, we also observe three key differences. First, developers need to use special tools such as Eclipse plug-in to use their techniques. In contrast, developers do not need to use any special tool for accessing code examples in eXoaDocs. Second, they try to provide as many code examples as they can find, which may include redundant usages, while, eXoaDocs cluster redundant usages into a cluster and present only the most representative one per each cluster using ranking. Third, eXoaDocs provide popularity information, which may guide developers find APIs even when they do not know the names APIs to use in advance.

Related to this popularity information, there are few existing systems that identify commonly used APIs [20], [14], [15]. For example, SpotWeb [20] finds commonly reused APIs and PopCon [14], [15] finds popular APIs based on API calls in existing software. While these systems only give popularity information, our eXoaDocs provide both popularity information and code examples for the APIs.

B. Code Search

Existing code search engines, such as Koders [7] and Google Code Search [1], get query keywords such as an API name and class name, retrieve source codes in a repository, and find source codes where the query keywords occur frequently. Because they treat source code as simple text and ignore semantic context of source code, they cannot distinguish useless information such as comments. As a result, they fail to find appropriate source code examples.

Meanwhile, there are some approaches to study semantic features of source codes. DeMIMA [12] and DECKARD [17] analyze semantic features of source code and find clones with similar semantic features. We adopt this idea of extracting semantic features in our work for generating good code examples and find similar code examples for finding different types of API usages.

V. Conclusion

Code examples play a major role in explaining the usage of APIs. However, as making code examples manually requires huge human effort, most API documents do not provide enough code examples. To address this problem, we proposed a technique that generates code examples automatically and embedded them into existing API documents, JavaDocs. Our evaluation results show that eXoaDocs embed source code examples for more than 20,000 APIs, or 75% of the entire APIs in JDK 5.

Our future work includes:

- *Comparing our code examples with other tools.* We need to compare the results of eXoaDocs with other tools for evaluating the quality of result code examples.
- *Performing user studies.* We need to conduct real-life user studies for evaluating how eXoaDocs affect the software development process.
- *Applying adaptive ranking algorithms.* We designed an adaptive ranking algorithm, and we are now collecting user feedback. We need to evaluate the quality of ranking adapting to these feedbacks.
- *Building a software repository.* Currently, our code example results depend on search results of Koders. Building a more general software repository to collect proper code example candidates will be the next step.
- *Precise analysis.* To find lines related with the given API, we used a simple method analysis technique by parsing source code. However, our technique sometimes failed to analyze the exact type of variables or classes due to the limitations reported in [16]. Overcoming these limitations will lead to higher quality code examples.

eXoaDocs generated for JDK 5 are accessible at http://exoa.postech.ac.kr.

VI. Acknowledgments

Our thanks to Yungbum Jung, Jinsun Her, Chris Bird, Mike Ernst and the anonymous reviewers of ASE for valuable and helpful suggestions on earlier versions of this paper.

This project is partially supported by the Engineering Research Center of Excellence Program of Korea Ministry of Education, Science and Technology (MEST) / Korea Science and Engineering Foundation (KOSEF), grant number R11-2008-007-03003-0.

References

[1] Google Code Search, 2009. http://www.google.com/codesearch.

[2] Java 2 Platform SE 5.0 API Specification: JavaDoc, 2009. http://java.sun.com/j2se/1.5.0/docs/api/.

[3] Java Examples (example source code), 2009. http://java2s.com/.

[4] Learn Java Programming by Examples, 2009. http://www.kodejava.org/.

[5] Leopard Reference Library, 2009. http://developer.apple.com/referencelibrary/index.html.

[6] MSDN Library, 2009. http://msdn.microsoft.com/en-us/library/default.aspx.

[7] Open Source Code Search Engine, 2009. http://www.koders.com.

[8] PHP: Hypertext Preprocessor, 2009. http://www.php.net.

[9] S. Bajracharya and C. Lopes. Mining search topics from a code search engine log. In *MSR 2009: 6th IEEE Working Conference on Mining Software Repositories*, New York, NY, USA, 2009. ACM.

[10] P. Devanbu, S. Karstu, W. Melo, and W. Thomas. Analytical and empirical evaluation of software reuse metrics. In *ICSE '96: Proceedings of the 18th international conference on Software engineering*, pages 189–199, Washington, DC, USA, 1996. IEEE Computer Society.

[11] J. E. Gaffney and T. A. Durek. Software reuse—key to enhanced productivity: some quantitative models. *Inf. Softw. Technol.*, 31(5):258–267, 1989.

[12] Y.-G. Guéhéneuc and G. Antoniol. Demima: A multilayered approach for design pattern identification. volume 34, pages 667–684, Piscataway, NJ, USA, 2008. IEEE Press.

[13] R. Holmes and G. C. Murphy. Using structural context to recommend source code examples. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 117–125, New York, NY, USA, 2005. ACM.

[14] R. Holmes and R. J. Walker. Informing eclipse api production and consumption. In *eclipse '07: Proceedings of the 2007 OOPSLA workshop on eclipse technology eXchange*, pages 70–74, New York, NY, USA, 2007. ACM.

[15] R. Holmes and R. J. Walker. A newbie's guide to eclipse apis. In *MSR '08: Proceedings of the 2008 international working conference on Mining software repositories*, pages 149–152, New York, NY, USA, 2008. ACM.

[16] S. Horwitz, T. Reps, and D. Binkley. Interprocedural slicing using dependence graphs. In *PLDI '88: Proceedings of the ACM SIGPLAN 1988 conference on Programming Language design and Implementation*, pages 35–46, New York, NY, USA, 1988. ACM.

[17] L. Jiang, G. Misherghi, Z. Su, and S. Glondu. Deckard: Scalable and accurate tree-based detection of code clones. In *29th International Conference on Software Engineering (ICSE)*, 2007.

[18] D. Kramer. Api documentation from source code comments: a case study of javadoc. In *SIGDOC '99: Proceedings of the 17th annual international conference on Computer documentation*, pages 147–153, New York, NY, USA, 1999. ACM.

[19] N. Sahavechaphan and K. T. Claypool. Xsnippet: mining for sample code. In *OOPSLA*, pages 413–430, 2006.

[20] S. Thummalapenta and T. Xie. Spotweb: detecting framework hotspots via mining open source repositories on the web. In *MSR '08: Proceedings of the 2008 international working conference on Mining software repositories*, pages 109–112, New York, NY, USA, 2008. ACM.

[21] T. Xie and J. Pei. Mapo: mining api usages from open source repositories. In *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories*, pages 54–57, New York, NY, USA, 2006. ACM.

[22] H. Zhong, T. Xie, L. Zhang, J. Pei, and H. Mei. MAPO: Mining and recommending API usage patterns. In *Proc. the 23rd European Conference on Object-Oriented Programming (ECOOP 2009)*, July 2009.