2012 19th Asia-Pacific Software Engineering Conference

An Exploratory Study of API Usage Examples on the Web

Lijie Wang, Yanzhen Zou, Lu Fang, Bing Xie, Fuqing Yang
Software Institute, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, P.R. China
Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871, P.R. China
{wanglj07, zouyz, fanglu09, xiebing, yang}@sei.pku.edu.cn

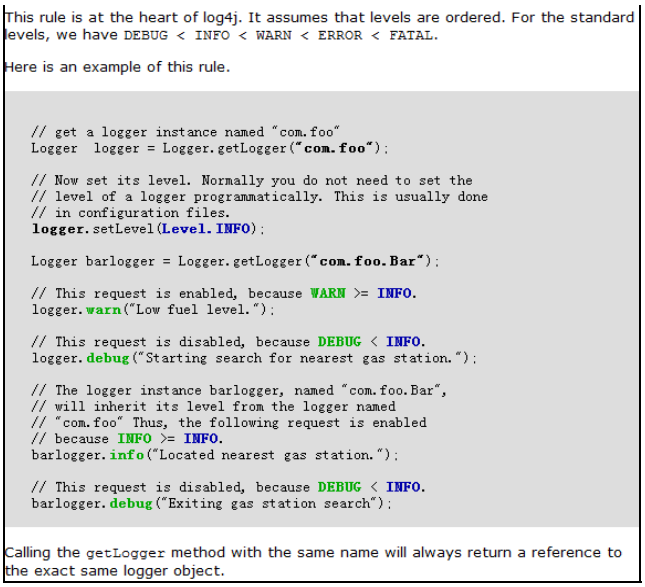
***Abstract*—Usage examples are helpful for programmers learning to use APIs from third-party frameworks or libraries. There are lots of usage examples scattered in web pages on the Web, such as tutorials, blogs, and forums. A few researches have proposed approaches to leveraging these usage examples to improve programming. However, due to the lack of comprehensive understanding on the current situation of usage examples on the web, the work is still at the very beginning. Many concerns are reserved, for instance, how many usage examples can be found on the Web? how well do such examples support programmers on earth? what factors have impact on these examples' usability? In this paper, we conducted an exploratory study of usage examples on the web, including their distribution, characteristics like content style, correctness, and complexity, as well as their correlations. Through the study, we obtain some insight of how to facilitate utilization of usage examples on the web and what mechanisms could be provided. Possible research directions and problems are proposed at the end.**

Keywords-empirical study, API, usage examples, web search

I. INTRODUCTION

Modern software industry increasingly relies on third-party libraries or frameworks (e.g., open source libraries). To use a library, programmers need to learn to use APIs in the library, but sometimes this can be very difficult [17][19][20]. In these cases, usage examples can be very helpful, as Albert Einstein said, "*Example isn't another way to teach, it is the only way to teach*". Learning by example has been widely adopted in software development [14][15][20][21][22]. Unfortunately, lots of libraries fail to provide sufficient usage examples in their documentations [2].

To address this problem, some approaches extracting usage examples from project codebase were proposed, such as [1][2][3][4][5]. Generally, they identify and slice code snippets which invoke target API from existing project source code. This provides a good starting point for getting examples from existing resources. However, because most existing projects are not initially designed for teaching, the code where target API is used often couples with the internal business logic tightly. This makes extracting code snippets from codebases may produce hard-to-understand results. In addition, the lack of descriptions about the examples impacts their results. The comments embedded in source code are generally about the implemented business logic from solution domain perspective. However, the programmer learning an API often needs descriptions from problem domain perspective to help her understand the example at hand better. The same issues are also exposed by existing code search engines (e.g., Koders).


This rule is at the heart of log4j. It assumes that levels are ordered. For the standard levels, we have DEBUG < INFO < WARN < ERROR < FATAL.

Here is an example of this rule.

```
// get a logger instance named "com.foo"
Logger  logger = Logger.getLogger("com.foo");

// Now set its level. Normally you do not need to set the
// level of a logger programmatically. This is usually done
// in configuration files.
logger.setLevel(Level.INFO);

Logger barlogger = Logger.getLogger("com.foo.Bar");

// This request is enabled, because WARN >= INFO.
logger.warn("Low fuel level.");

// This request is disabled, because DEBUG < INFO.
logger.debug("Starting search for nearest gas station.");

// The logger instance barlogger, named "com.foo.Bar",
// will inherit its level from the logger named
// "com.foo" Thus, the following request is enabled
// because INFO >= INFO.
barlogger.info("Located nearest gas station.");

// This request is disabled, because DEBUG < INFO.
barlogger.debug("Exiting gas station search");
```

Calling the `getLogger` method with the same name will always return a reference to the exact same logger object.

Figure 1. A typical usage example from web page (This is a usage example of API `org.apache.log4j.Logger`. This is a screenshot of a part of web page at: http://logging.apache.org/log4j/1.2/manual.html)

With the development of Web 2.0, Internet has been a universal platform for communication between human beings, including programmers. More and more programmers become used to sharing their programming experience and lessons on the Web. The vast programming information published on the Web, e.g., tutorials, technical blogs, Q&A threads, and forums posts, provides a rich resource of usage examples. ***In a web page, a usage example refers to a code snippet and its surrounding related texts.*** Fig. 1 demonstrates a typical usage example from a web page, including a code snippet accompanied with two segments of descriptive texts. In this paper, we refer to "usage examples on the web" as "usage examples in web pages" and we will routinely use these terms interchangeably for the same meaning.

Existing observations indicate that many programmers rely on such usage examples on the web to learn to use APIs [8][15][21]. However, one of the main problems faced by programmers when searching for these usage examples with traditional web search tools (e.g., Google) is that they often need to spend great effort in finding out relevant examples from vast data in lots of web pages due to the existence of irrelevant noises, scattering results, and data duplication

1530-1362/12 $26.00 © 2012 IEEE
DOI 10.1109/APSEC.2012.122

[8][21]. To help programmers use examples on the web, several researches were proposed [7][8][9][10]. The main idea is collecting web pages which contain sample code, and then extracting examples from pages using some strategies. Usage examples are finally delivered to programmers via a unified view. However, existing work treat all usage examples on the web equally, although they can differ with each other. For instance, usage examples from a tutorial page might be different with those from a forum page in some aspects, such as content, correctness, and context. In addition, proponents claim that there are prevalent usage examples on the web and they can be used to help programmers. But we are not aware of any empirical study dedicated in such claim.

While there are some studies of development related information on the web and of user behavior in code retrieval on the web, few researches are dedicated to usage examples on the web. To find out what characteristics usage examples on the web have，how well do they support programmers, and what can be done to utilize them better so as to improve API based programming, we conducted an exploratory study of the current situation of usage examples on the web with qualitative and quantitative approaches. Through the study, we obtain some insight of usage examples on the web. The study results indicate that, for most APIs, considerable good usage examples can be found on the web. However, some examples with low usability exist, too. Correctness, readability, and source type are some factors impact example's usability. For instance, we find that programmers prefer examples with good readability, examples from discussion style pages (e.g., forums and Q&A) have more correctness issues. Based on the study, we propose some possible research directions and problems which might be worth being invested to improve the state of the art, expecting to provide support for scholars in related areas.

The rest of this paper is organized as follows: in Section II, we present the work related to this study. Section III presents research questions we are concerned. The study methodology is introduced in Section IV. Section V presents results of the study, and Section VI discusses findings according to the study results. Limitations of our study are discussed in Section VII, and conclusions are drawn in Section VIII.

II. Related Work

With the development of Web technology, almost everything can be accessed through the World Wide Web, including software development related artifacts and information. Researches related to our work are mainly empirical studies on such information on the web and their effecting roles in programming.

Treude *et al.* conducted a series of study on roles of social media in software development (*community portals* in [12], *Q&A sites* in [13], and *blogs* in [11]) and discussed possible directions in which improvements can be made. In their study, they concentrated on studying a specific media site each time while our study focuses on a specific type of software development related information, i.e., usage examples. These researches can be complementary to each other.

Susan *et al.* conducted an in-depth study on how well do existing search tools (e.g., general purpose search engine, code search engine, and software reuse system) support code retrieval on the web [15][18]. They find that although a general-purpose search engine is the most effective overall on all tasks, code-specific search tools are still needed. In addition, their study results indicate that searching for reference examples had been largely overlooked by the work on reuse.

Brandt *et al.* studied programmers' information needs via observing their activities in web foraging, learning, and coding [21]. They find that opportunistic programmers intensively rely on the online resources (especially usage examples) to learn to programming. Based on the study, they proposed a search interface in IDE allowing programmers directly carry out example search in programming environment [9]. Usage examples are delivered to programmers via a code-centric view. Similar with Brandt's work, Hoffmann *et al.* identify code snippet from web pages, then use the implicit references among code, JAR files and API documents to support common programming search task [8]. APIExample proposed by us collects usage examples from the Web and conduct some statistic analysis and clustering on the examples [7]. Such attempts are good starting point to facilitate the utilization of usage examples on the web. However, in these researches, the characteristics of online examples are not studied thoroughly.

In [19] and [20], Hou *et al.* and Robillard *et al.* studied the barriers faced by programmers while using existing APIs, respectively. They shared some valuable implications for guiding future APIs and documentations design. They claimed that usage example should be provided to demonstrate "best practice" for using an API. In their study, they find that programmers who retrieved example from the web are sometimes suspicious of examples' quality, and it is important to estimate the quality of examples on the web. The results of our study can provide implications to achieve this task.

III. Research Questions

To obtain a comprehensive understanding of usage examples on the web, we formulate research questions within three categories: 1) *quantity related questions*, 2) *quality related questions*, and 3) *effect related questions*.

1) How many usage examples are there on the web?

Given a library, how many APIs have usage examples on the web? How many usage examples do they have? Where can usage examples be found?

2) What characteristics do usage examples on the web have?

Being aware of characteristics (especially the special ones) of usage examples on the web could help us figure out targeted problems and approaches to utilize the examples better. In this study we mainly focus on the following characteristics: correctness, complexity and readability.

3) How well do usage examples on the web support programmers for learning APIs?

Can programmers learn to use APIs through studying usage examples on the web? What kind of usage examples

do programmers prefer? What characteristics do these examples have?

IV. METHODOLOGY

To answer the research questions, we divide the study into two steps, data collection and data analysis. In data collection, we invent some automated approach to harvest usage examples from the web. Based on the collected data, in data analysis, we leverage human effort and machine to analyze the data, respectively.

A. Data Collection

As we know, it is infeasible to collect all of the usage examples on the web for all APIs. To conduct the study, we need to select some APIs as representative. In this study, we use five famous Java open source libraries as study subjects, including JDK, JDOM, Log4j, Lucene, and Apache Commons DBUtils. We collect usage examples for APIs in these libraries. Scale of the five subject libraries is listed in Table I.

For each API in the libraries, we use an automated approach proposed by us [7] to collect its usage examples from the web. The approach contains the following steps:

(1) Leverage Google search engine to collect web pages containing usage examples from the Web. To collect usage examples from the Web, we need to get web pages containing them first. Given an API, we leverage Google to collect its related web pages from the Web through constructing query in format of `"API_FQN" example java`, where `API_FQN` is the full qualified name of the given API, terms `example` and `java` are used to restrict the searching scope. The top N (currently 200) web pages in the results list are downloaded as the API's related web pages. URL and content of each web page are recorded. The correspondence relationships between web pages and APIs are also recorded in database.

(2) Extract usage examples from web pages. As mentioned in Section I, in a web page, we refer to a usage example as a code snippet and its surrounding related texts. According to this principle, we leverage a surface-feature-and-incremental-parsing-based approach to automatically identify and extract usage examples from each collected web page. All of the extracted usage examples and the relationships between examples and their original web pages are recorded in database.

TABLE I. SCALE OF SUBJECT LIBRARIES

Library	Version	# APIs	Homepage
JDK	1.6	3,390	http://www.oracle.com/technetwork/java/javase/downloads/index.html
JDom	1.1.1	51	http://www.jdom.org/
Log4J	1.2	193	http://logging.apache.org/log4j/1.2/
Lucene	2.9.4	981	http://lucene.apache.org/
DBUtils	1.3	22	http://commons.apache.org/dbutils
Total		**4,637**	

(3) Identify reference relationships between usage examples and APIs. In this step, we use an adaptable parser to parse the code snippet of each usage example to identify which APIs the example references. If a usage example references an API (e.g., invokes the API's method, accesses the API's field), we record it as an example of the API. An API might have some usage examples while a usage example might reference some APIs. The correspondence relationships between usage examples and APIs are recorded in database.

B. Data Analysis

We analyze the collected data using a mix of qualitative and quantitative approach. For instance, based on the data stored in database, we use some statistics scripts to calculate how many usage examples an API has, how many APIs a usage example references. Regarding some characteristics such as correctness and usability, we invited some programmers (called "judger") to label manually. Table II lists approaches for measuring different metrics.

TABLE II. MEASURING APPROACHES FOR DIFFERENT METRICS

Measuring Approach	Metrics	Explanation
Statistics Script	Example Coverage	*How many usage examples each API has on the web.*
	Complexity	*The complexity of a usage example, such as line of code (LOC), how many APIs a usage example references.*
Manual Labeling	Source Type	*Which type of web pages a usage example comes from, such as blog and forums.*
		Whether a usage example comes from the official web site of its referenced API.
	Usability	*Whether a usage example useful for an API learner.*
	Readability	*Readability refers to the ease in which a usage example can be read and understood.*
	Correctness	*Whether the usage example demonstrates the right usage of related APIs.*

In order to ensure reliability of manual labeling, each judger needs to be familiar with the APIs he/she labels. For the five subject libraries, we totally recruited 10 senior graduate students from our laboratory as judgers, and compensated them with a 50RMB gift card each. They all have rich experience of more than 3 years Java development. In the recruiting, we asked each judger candidate about their development experience to identify which libraries he/she is familiar with. All of the recruited graduates are familiar with JDK APIs; 4 graduates are familiar with Log4J; 2 graduates are familiar with JDom, Lucene, and DBUtils, respectively. In Table III, a blank diamond indicates that the corresponding judger is familiar with the corresponding library. According to judger's profile, we assign a subject library to the programmers who are familiar with it. In Table III, a black diamond indicates that the corresponding judger is assigned to label API usage examples for the corresponding library. For instance, judger J5 and J6 are assigned to label API usage examples for JDOM, because their profiles indicate that they are familiar with the project.

Considering that much human effort need be spent in labeling, for each subject library, we provide an API pool from which judgers can select some APIs to label. From each

library, we select top 20 APIs which have the most examples to construct the API pool. Given a selected API, the labeling system randomly chooses some related usage examples to label. For each usage example, the judger needs to label items in the cells with gray background in Table II. If the judger has very low confidence in correctly labeling the example, he/she can select to skip it and forward to the next one. For each labeled example, our labeling system ensures that there are at least two judgers labeling it. If the labeling results of the two judgers have conflict, a third judger will be assigned to make the decision. The numbers in brackets in Table III represent number of APIs in the corresponding library labeled by the corresponding judger. For instance, J1 and J2 both labeled the same 7 APIs of JDK library. Finally, there are totally 34 APIs' 2,031 usage examples from 1,159 web pages (listed in Table IV) are labeled.

TABLE III. LABELING TASK ASSIGNMENT

Judger \ Lib	JDK	Log4J	Lucene	JDOM	DBUtils
J1	◇◆(7)	◇◆(4)			
J2	◇◆(7)		◇◆(6)		
J3	◇◆(8)				
J4	◇◆(8)				
J5	◇			◇◆(3)	
J6	◇	◇		◇◆(3)	
J7	◇	◇			◇◆(6)
J8	◇	◇◆(4)			
J9	◇				
J10	◇		◇◆(6)		◇◆(6)

TABLE IV. APIS LABELED BY JUDGERS

Lib	API	Lib	API
JDK	java.awt.BorderLayout	DBUtils	org.apache.commons.dbutils.DbUtils
	java.awt.Color		org.apache.commons.dbutils.handlers.ArrayHandler
	java.awt.Component		org.apache.commons.dbutils.handlers.ArrayListHandler
	java.awt.Container		org.apache.commons.dbutils.handlers.BeanListHandler
	java.awt.Dimension		org.apache.commons.dbutils.QueryRunner
	java.awt.event.ActionEvent		org.apache.commons.dbutils.ResultSetHandler
	java.io.File	Log4J	org.apache.log4j.BasicConfigurator
	java.io.FileInputStream		org.apache.log4j.Layout
	java.io.FileOutputStream		org.apache.log4j.Level
	java.io.InputStream		org.apache.log4j.Logger
	java.net.URL	Lucene	org.apache.lucene.analysis.Analyzer
	java.sql.ResultSet		org.apache.lucene.document.Document
	java.util.Date		org.apache.lucene.index.IndexReader
	java.util.List		org.apache.lucene.queryParser.QueryParser
	javax.swing.JButton		org.apache.lucene.search.BooleanQuery
JDOM	org.jdom.Attribute		org.apache.lucene.search.IndexSearcher
	org.jdom.Document		
	org.jdom.input.SAXBuilder		

V. STUDY RESULTS

We perform analysis based on the statistic results and labeled data, and give answers to the research questions.

A. How many usage examples are there on the web?

1) Distribution of Example Numbers

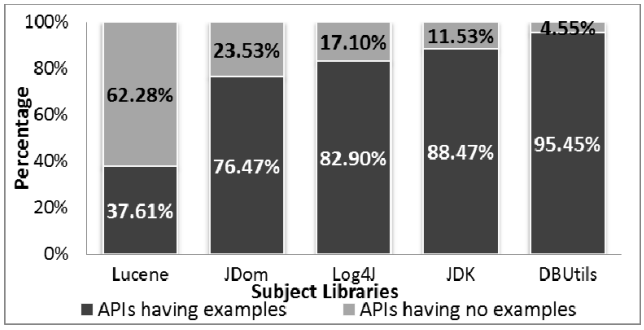


Figure 2. Proportion distribution of APIs having usage examples

We finally collected 322,040 distinct web pages among which 36,391 (11.3%) pages contain API usage examples[1]. The ratio indicates that, on average, a programmer can only find one usage example by browsing nearly 10 web pages from Google's search results. 61,387 usage examples were extracted from the 36,391 pages. ***On average, 77% of the APIs in subject libraries have related usage examples on the Web.*** 699,720 API-example reference relationships were identified. Each API is averagely referenced by 196 usage examples. All subject libraries, except Lucene, have over 75% APIs having usage examples, as shown in Fig. 2. Nearly 90% APIs of JDK have usage examples on the web; only 1 API of DBUtils has no usage examples on the web. We inspect Lucene for the reason of low example coverage ratio, find that most APIs in Lucene are designed for internal-use (i.e., used by other APIs in Lucene). Among the APIs designed for external-use, the ratio of example coverage is relatively high. For instance, in Lucene's `indexing`, `document`, and `searching` packages[2], the ratio of APIs having usage examples grows up to 61%.

TABLE V. HOTSPOTS DETECTED WITH NUMBER OF EXAMPLES ON THE WEB (APIS IN GRAY CELLS ARE HOTSPOTS DETECTED BY SPOTWEB) COLUMN 'RANK' IS API'S ORDER ACCORDING TO NUMBER OF ITS USAGE EXAMPLES COLLECTED FROM THE WEB.

Rank	API	#Examples
1	org.apache.log4j.Logger	1367
2	org.apache.log4j.Category	1301
3	org.apache.log4j.spi.LoggingEvent	419
4	org.apache.log4j.Level	342
5	org.apache.log4j.PropertyConfigurator	328
6	org.apache.log4j.BasicConfigurator	218
7	org.apache.log4j.Layout	211
8	org.apache.log4j.Priority	204
9	org.apache.log4j.helpers.LogLog	191
10	org.apache.log4j.PatternLayout	130
11	org.apache.log4j.AppenderSkeleton	129
12	org.apache.log4j.xml.DOMConfigurator	124
13	org.apache.log4j.LogManager	110
14	org.apache.log4j.spi.LoggerRepository	108
15	org.apache.log4j.Appender	96
16	org.apache.log4j.FileAppender	85
17	org.apache.log4j.ConsoleAppender	82
18	org.apache.log4j.chainsaw.Main	80
19	org.apache.log4j.Hierarchy	76
20	org.apache.log4j.spi.ErrorCode	67

[1] Data collection was carried out on Feb. 20th, 2012.

[2] Since Lucene is a text retrieval library, its indexing and searching packages are frequently used by other software systems.

In addition, we find that ***the number of an API's usage examples collected from the web could reflect the API's popularity degree.*** In SpotWeb [6], Thummalapenta *et al.* identify *hotspots* (*popular APIs*) and *codespots* by mining code examples gathered from open source codebase. Hotspots can be used as starting points for programmers in reusing libraries while codespots can be used as caveats for programmers as there can be difficulties in finding relevant examples and are generally less exercised compared to hotspots. We compared the hotspots of Log4j detected by their approach with top APIs having most usage examples from the web and find that, as shown in Table V, 10 of the 11 hotspots APIs detected by SpotWeb appear in the 20 top ranked APIs, the top 10 APIs ranked by our approach contain 7 hotspots.

2) Usage Example Sources Distribution

There are many types of web pages on the Web, such as technical blogs, tutorials, and forums. The styles and content in different web pages differ with each other. Usage examples from different type of web pages might have differences, too. Being aware of examples' source types may help us find targeted research for some specific sources.

We divide source web pages of usage examples on the Web into following nine types:

- **API Documents**. Structural documents of API interfaces, such as Javadoc of Java API. Some API documents are hosted on the API's official sites while some documents might be hosted on some third-party sites. Here is an example of API documents: `http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html`
- **Technical Articles**. A piece of writing that introduces solutions to some problems or the usage of some APIs, such as technical blogs, tutorials, and user manuals. Technical blogs are typically published by some third-party programmers and organizations to share their programming experience and lessons. Tutorials are typically used to talk about the usage of some frameworks or APIs systematically. Here are some examples of Technical Articles: `http://blogs.msdn.com/b/sergeim/archive/2009/04/00/unzip-file-via-java-util-zip-package-from-net-code.aspx`, `http://silveiraneto.net/2010/02/27/the-caps-lock-java-socket-server/`
- **Forums**. Technical forums where programmers exchange ideas and discuss issues about programming. Here is an example of Forums: `http://www.java-forums.org/new-java/13248-buffered-reader-exception.html`
- **Q&A**. Web sites where programmers ask and answer questions about programming. The structure of Q&A sites is similar to that of Forums. They are both in the form of discussion threads. But the topic in one thread of Q&A is typically more focused. Here is an example of Q&A: `http://stackoverflow.com/questions/621117/how-do-i-fix-this-java-generics-wildcard-error`
- **Sample Sharing Sites**. Web sites which are used to share sample code. There are typically only one or some segments of sample code in the web pages. Some sample code might be organized manually (labeled as *Sample_Manual*) while some might be generated by machine (labeled as *Sample_Tool*). Here are some examples of sample sharing sites: `http://www.java2s.com/Tutorial/Java/0320__NetworkNe/javanetURL.htm`, `http://www.javadocexamples.com/java/io/BufferedReader/reset().html`
- **Snippet Sites**: Web sites used to store text snippets for memory or sharing, such as pastebin.com.
- **Source Code**. Some web pages containing a full code file which are typically returned by code search engine or online version control system. Here are some examples of source code pages: `http://cr.openjdk.java.net/~mullan/webrevs/6994717/webrev.00/test/java/security/cert/CertPathValidator/nameConstraintsRFC822/ValidateCertPath.java-.html`
- **Mailing Lists**. Mailing list archive such as `http://www.kaffe.org/pipermail/kaffe/1999-June/174220.html`
- **Issue Tracking System**. Archive hosted on the issue tracking systems which can be accessed online. Here is an example of Issue Tracking System: `http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4491595`

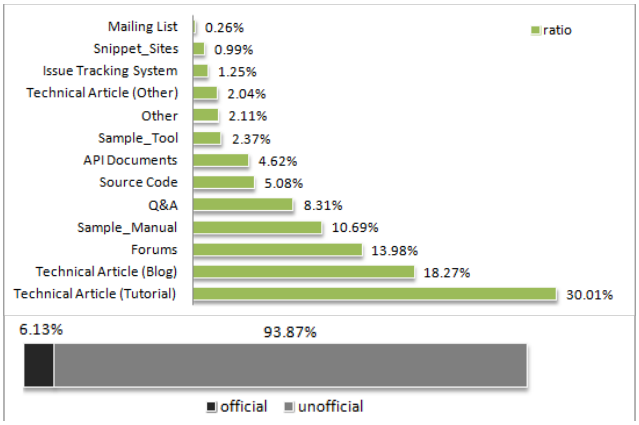


Figure 3. Distribution of usage example's source web sites

Based on the taxonomy above, we let judgers label usage examples' source types. The distribution of usage example's source type is shown in Fig. 3. From the results we can find that nearly 50% usage examples come from Technical Article type web pages, especially tutorial (30%) and blog (18.3%). The second largest source type is discussion style pages, including Forums (14%) and Q&A (8.3%). There are about 13% usage examples come from Sample Sharing web sites. Other 17% usage examples come from JavaDoc (4.6%), Source Code (5%), Mailing List (0.3%), Issue Tracking System (1.25%), and so on.

The result indicates that ***usage examples on the web are scattered across many different types of web sites***. However, ***Technical Article (tutorial, blog) and Discussion Style (Forums, Q&A) web pages occupy the largest number of usage examples***.

Among the collected usage examples, nearly 94% examples come from unofficial web sites; only 6% examples come from official web sites. Because organizing examples manually will cost much effort and time, most API providers cannot provide usage examples for all APIs. This indicates that ***it's impractical for an API learner to find enough usage examples from the API's official site***. In addition, there are only 5% examples come from API Documentations.

Therefore, it is necessary to enrich usage examples from more data sources to ease API learning barriers.

B. What characteristics do usage examples on the web have?

Usage examples on the web are referred to as a sample code segment and its surrounding related texts embedded in a web page. Compared with usage examples gotten from other sources (e.g. project codebase), examples on the web might have some special characteristics which can introduce some targeted mechanisms or researches on the topic of utilization of such examples. In this section, we will explore this topic in detail.

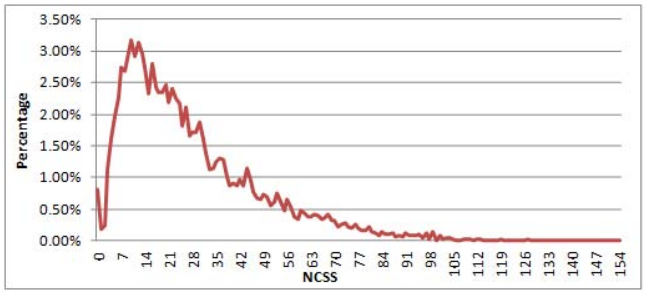
Regarding characteristics, we mainly concern the following aspects:

Complexity: Line of sample code in a usage example; Number of APIs referenced by each usage example.

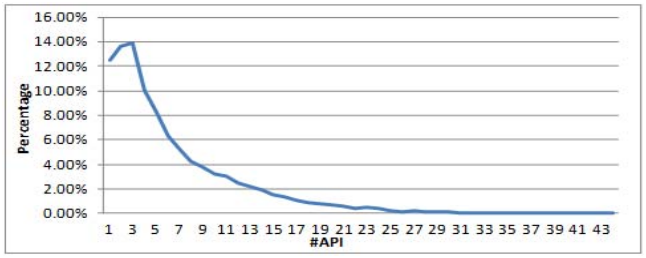
Readability: Amount of descriptive texts each usage example has; Displaying style in original web pages.

Correctness: Whether the examples demonstrate the right usage of APIs; Completeness of the depended context.

1) Complexity of Usage Examples on the Web



(a) Distribution of NCSS in each example's code snippet



(b) Distribution of number of APIs referenced by each example

Figure 4. Distribution of usage example complexity

We conducted statistics analysis on number of *None Commenting Source Statements* [3] *(NCSS)* in each usage example's code snippet. The distribution is shown in Fig. 4(a). The NCSS of most usage examples fall into region 10~50 with proportion of 70%. There are 16% usage examples' NCSS less than 10. The proportion of usage examples with NCSS larger than 80 is only 2.3%.

The distribution of *Number of APIs each example references* is shown in Fig. 4(b). There are over 60% usage examples invoking 2~8 APIs. The proportion of usage examples which reference over 15 APIs is only 10%.

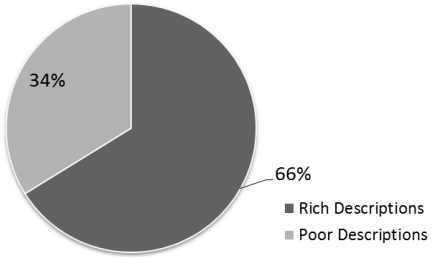
The results indicate that ***the complexity of usage example on the web is in medium level***. As claimed by Robillard *et al.* in [20], the pedagogical power of examples is perceived to decrease as the complexity of the examples grows. From this perspective, usage examples in medium size might be friendly to learners.

2) Readability of Usage Examples on the Web

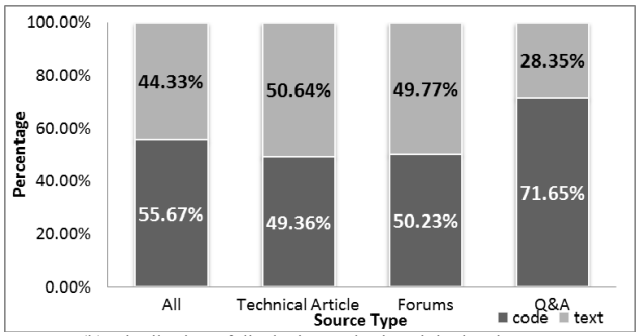
Readability refers to the ease in which a usage example can be read and understood. Here we use two metrics to measure this characteristic, i.e., *how many descriptive texts each usage example has*, and *the displaying style of usage example in original web page*.

We ask judgers to judge whether the usage examples have rich descriptive texts (i.e., the descriptive texts can be helpful for understanding the corresponding example). The distribution is shown in Fig. 5(a). Meanwhile, we provide URL of example's original web page with which judgers can visit the web page to judge examples' original displaying style (displayed as plain texts or displayed as structural code). The distribution is shown in Fig. 5(b).

From the result in Fig. 5(a), we can find that most (nearly 70%) usage examples from the web have rich descriptive texts. This indicates that ***most usage examples can be easy to be understood by API learner***. Although usage examples from the web have rich descriptions, as shown in Fig. 5(b), ***about half of them are displayed as plain texts in their original web pages.*** Such displaying style can make examples difficult to be read and understood.



(a) Distribution of richness of descriptive texts



(b) Distribution of displaying styles in original web pages

Figure 5. Distribution of usage example readability

3) Correctness of Usage Example on the Web

Correctness is a crucial characteristic of usage example. It impacts the usability and reference-value heavily. Therefore, we try to capture a view on the correctness of

[3] We use a source measurement suite for Java named as JavaNCSS (http://www.kclee.de/clemens/java/javancss/) to conduct the analysis.

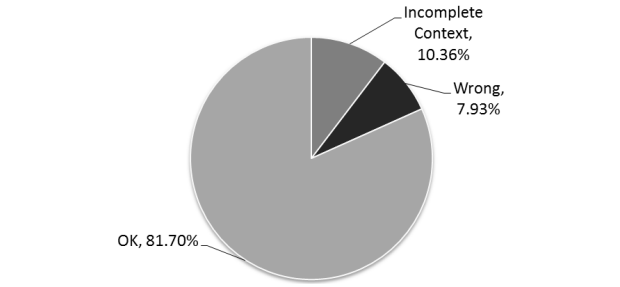
usage examples from the web. We divide correctness situation into three types:

OK: A usage example demonstrates the usage of related APIs in a right way, and accomplishes the declared function.

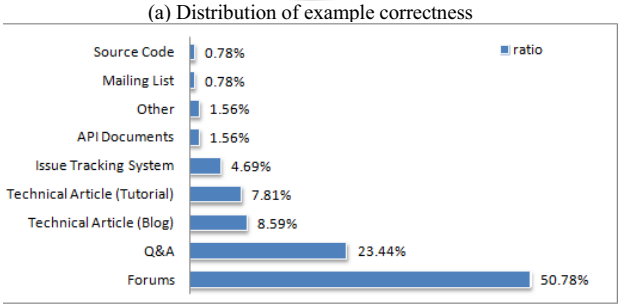
Incomplete Context: The depended context of a usage example is incomplete. For instance, the declaration of some variables referenced by the usage examples is not presented in the example's code.

Wrong: The usage example has mistakes in the usage of related APIs, such as passing wrong parameters, invoking improper methods. Most wrong examples are introduced unintentional (e.g. carelessness) or published to ask for help (e.g. on forums).

We ask judgers to judge the correctness of usage examples according to example content and surrounding environment, as well as their experience. They can test examples by executing them in IDE if needed. The distribution of example correctness is shown in Fig. 6(a). From the result we can see that most (about 82%) usage examples are correct. About 18% usage examples have problems in correctness. More in-depth, we analyze the source distribution of the examples whose correctness is not "OK". As shown in Fig. 6(b), over 50% and 23% incorrect examples come from Forum and Q&A pages, respectively.



(a) Distribution of example correctness



(b) Distribution of source types of "not-OK" examples

Figure 6. Distribution of usage example correctness

The results above indicate that, on the one hand, ***many correct usage examples can be found on the web***; on the other hand, ***how to judge example's correctness need to be paid special attention to*** during web based examples collection, especially collecting examples from Forum and Q&A sites.

C. How well do usage examples on the web support programmers for learning APIs?

There is a large number of API usage examples scattered on the Web. These examples can be used by programmers to learn to use APIs. However, how well do such usage examples support programmers for learning APIs on earth? What factors have impact on example's usability? In this section, we try to answer these concerns through some statistics and analysis.

To achieve this objective, we ask judgers to judge examples' usability according to his/her programming experience through rating on the examples. We let them judge the usability from the perspective of an API learner assuming that they want to learn to use APIs through studying the example. Rating 5 means that the usage example demonstrates the usage of related API perfectly. Rating 0 means that API learner completely cannot learn to use the API through the example.

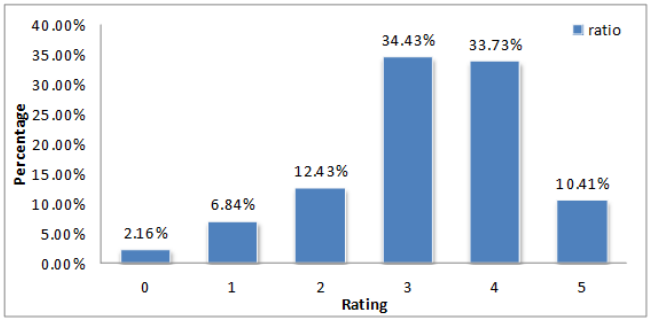


Figure 7. Distribution of usage example's rating

The distribution of example usability is shown in Fig. 7. We can see that about 44% examples receive rating 4~5 (high rating), while 34% examples have rating 3 (medium rating). There are 21% examples receive rating 0~2 (low rating). Considering that the number of usage examples on the web is very large, this result indicates that ***considerable number of usage examples with good usability can be found on the web***. Of course, ***there are some examples with bad usability on the web, too***. To utilize the usage examples on the web effectively, we need to find out what factors have impact on the usability of examples. We consider this issue from three aspects: example's sources, example's correctness, and example's readability.

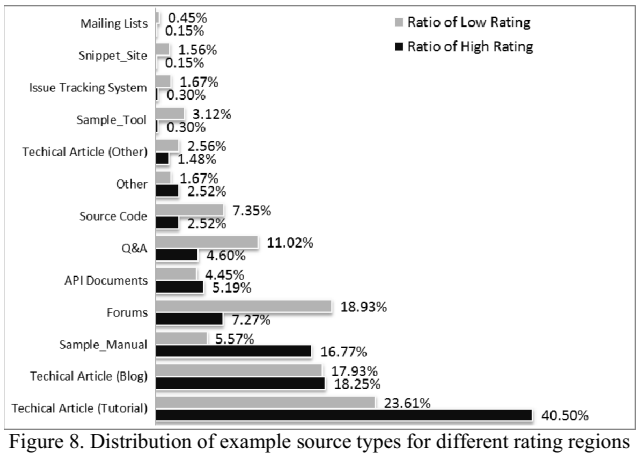


Figure 8. Distribution of example source types for different rating regions

In Fig. 8, the black blocks and the gray blocks illustrate source distribution of examples with rating 4~5 and source distribution of examples with rating 0~3, respectively.

Through comparison between the two distributions we can find that, the proportion of examples from Forums and Q&A pages in the low-rating-distribution is significantly higher than that in the high-rating-distribution (increase from 11.87% to 29.95%, the growth ratio reaches up to 152%). In addition, another source type with high rate of increase is Source Code (increases from 2.52% to 7.35%). This indicates that, ***results returned by source code search engines are actually not as ideal as programmers want***. The comparison also indicates that ***source type of usage examples may play as an important factor for judging example usability***.

We analyzed the correlation between example's usability and whether the example comes from official site. As shown in Fig. 9, programmers obviously prefer usage examples from official sites. This indicates that ***the examples from official sites should be given higher priority in example recommendation or retrieval***. To some extent, this is intuitive because programmers are generally concern about the authority and credit of examples [20].

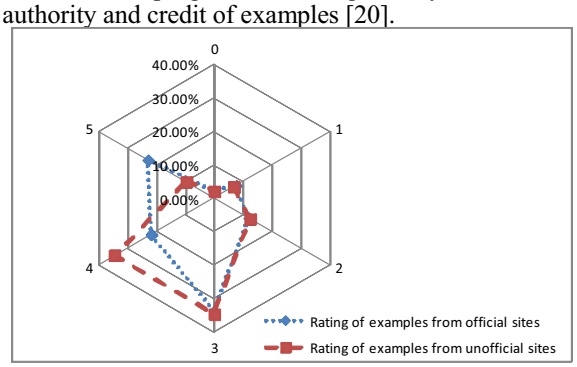


Figure 9. Correlation between example's rating and its officail/unofficial source types

Regarding the correlation between example's usability and its correctness, Fig. 10 illustrates that *an example's* ***correctness situation has positive correlation with its usability***. Programmers prefer usage examples with less correctness problems.

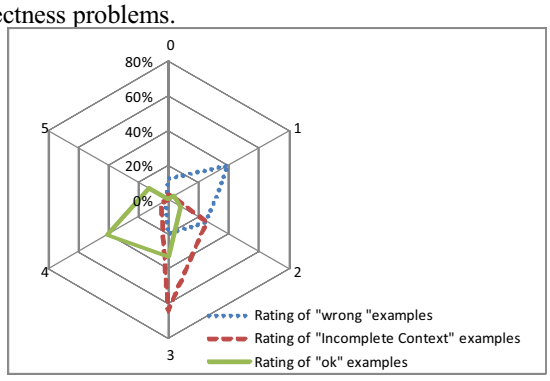
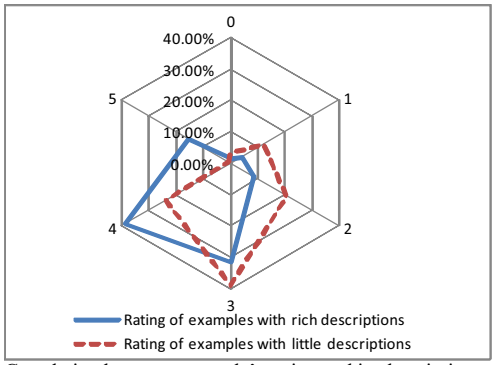
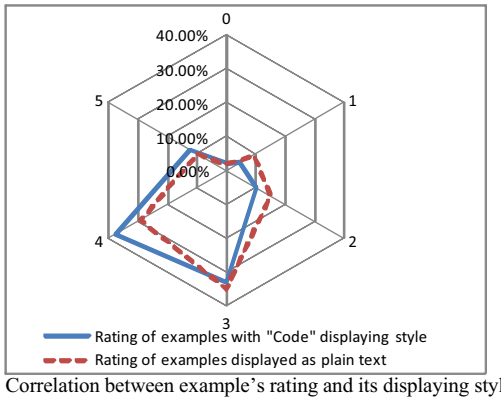


Figure 10. Correlation between example's rating and its correctness

Regarding readability, we analyzed the correlation between example's usability and its displaying style in original web page as well as its amount of descriptive texts. The results are shown in Fig. 11(a) and 11(b), respectively. We can see that ***the better an example's readability is, the higher rating it receives***.



(a) Correlation between example's rating and its descriptive texts



(b) Correlation between example's rating and its displaying style

Figure 11. Correlations between exmaple's rating and its readability

VI. FINGDINGS

Base on the study results, we draw some findings with the expectation of supporting related researches in the future.

A. *Web pages are good resources of usage examples*

As the results indicate, ***nearly 80% APIs have usage examples on the web. And the number of examples APIs have is considerable.*** The high coverage makes it possible for programmers to learn to use most APIs with examples on the web. Usage examples generally have relatively rich description texts making them easier to be understood. Therefore, the web pages containing programming information on the Web provide a good resource for getting usage examples.

B. *Mechanisms need be provided to support programmer use exmaples on the web*

Although there are a large number of usage examples on the Web, programmers will face many problems while searching and using these examples. Examples distribute across lots different types of web sites, programmers need to browse everywhere to find the required examples. About half examples in web pages are displayed as plain texts, this brings burden to programmers in viewing examples [20]. In addition, although programmers prefer examples with less correctness issues, there are considerable incorrect examples on the web. Therefore, ***researches for helping programmers use examples on the web easily and effectively are needed***. For instance, precisely collect examples from the web and

deliver them to programmers in a unified interface. Of course, many challenges (either technical or cultural) will be encountered in this process. We hope the results of this study could provide some support for figuring out research problems and for proposing solutions.

1) ***Selecting high-quality usage examples***

Since there are so many usage examples on the web, it is necessary to select high-quality ones (or filter out low-quality ones). The study result indicates that programmers prefer examples with less correctness issues. Therefore, in examples collection, ***correctness of usage examples could be used as a criterion***. Of course, automatic figuring out which examples have correctness issues is not obvious. According to the study results, we think there are some information can be used as references, for instance, the source types. The correlation between example's correctness and source type indicates that, ***examples from discussion style web pages have relatively more correctness issues***. The context of usage examples can also be used as a criterion, too. A usage example whose context contains clues like "exception" "error" "unexpected" etc. is more likely to be incorrect. In addition, traditional techniques used for bug detection in software system can be used. For instance, existing API specifications [23] could be used to detect potential correctness issues.

In addition to correctness, there are some other characteristics could be used for quality evaluation, including source type, amount of description texts, as well as the example's displaying style in the original web page. ***Examples from official sites should be more reliable. Programmers generally prefer examples with rich description texts. And the examples which are originally displayed in code style receive high ratings.*** Of course, how to automatically measure these characteristics need be explored. Moreover, to identify "good" examples, techniques for finding high-quality content in information retrieval area like [16] might be borrowed and adapted.

2) ***Mining Usage Examples from the Web***

Usage examples from the web contain rich information for mining. For instance, API specification might be mined from usage examples in the systematic content of blog or tutorial style web pages. API's frequently encountered issues can be mined by analyzing discussion archive in forums and Q&A sites. In addition, as mentioned in Section V-A-1, API's popularity can be estimated via mining API's usage examples on the web.

3) ***Combining usage examples from the web and usage examples from codebase***

Usage examples from the web have rich descriptions but have correctness issues, while examples from codebase have little correctness issues but have few descriptions. ***It is possible to combine usage examples from the web and those from codebase by leveraging their advantages to overcome mutual shortcomings.*** In addition, rich descriptions of usage examples from the web can be used to bridge the gap between problem domain and solution domain which is an important issue for code retrieval [15] [19] [20].

VII. DISCUSSIONS AND LIMITATIONS

A. Repsentativeness of sample data set

In our study, we examine usage examples on the web from the perspective of APIs in five java open source libraries (JDK, Log4j, Lucene, DBUtils, and JDOM). Although this provides a good starting point, but there might be limitations on the representativeness of study subjects. The prevalence of usage examples on the web for Java APIs might be different from that for APIs in other programming languages. Moreover, since the five subject libraries are all from relatively famous and long-history projects, their usage examples on the web might be more than other libraries. It is a better choice to conduct larger scale study on usage examples for APIs in different languages and different kinds of projects. We plan to carry out some concrete researches in this direction. Nevertheless, this study results reflect that it is practical to gather usage examples from the web for widely-used APIs.

B. Limitations of the data collection approach

Usage examples can be everywhere on the web. For a research institute, it is impossible to build a super-crawler to harvest examples all over the web. In this study, we leverage Google to collect examples from the web through constructing search queries with API's FQN (Full Qualified Name). Since commercial search engines index most resources all over the web, it is possible to get the required data if we can tell the search engine what we want exactly. However, there might be some limitations about the queries we constructed. Some usage examples might reference an API without explicitly listing its FQN. In addition, we add keywords '`example`' and '`java`' to restrict search scope. We only record URLs in the top 200 search results, and this might reduce the search results amount. However, with the improvement of collection approach, the number of API usage examples on the web should increase. In the data collection, we did not handle differences between different API versions. Actually, it is also a big issue to be conquered in web based example collection.

C. Reliability of Manual Labeling

In this study, we rely on manual effort to figure out some characteristics of usage examples. Due to the subjectivity of manual effort, bias may be introduced into the study. To avoid such impact, we invited experienced programmers to label examples for APIs which they are familiar with. In addition, cross-validation is used when determine labeling results. Since the essential value of usage examples is that they can help programmers to study APIs better, human factor is inevitably. To improve the result's reliability, some programming tasks can be assigned to programmers and observe their activities while they programming with examples on the web. In addition, some stricter and more objective criteria could be proposed.

D. Limitations on Findings

Based on the data collected, authors of this paper draw some findings in terms of usage examples on the web

according to their knowledge background in related area. Due the limitations of breadth and depth of personal knowledge accomplishment, some constructive findings and interesting results might be neglected. We hope that other scholars can conclude more findings based on the data.

VIII. CONCLUSION

In this paper, we conducted an exploratory study on the current status of usage examples on the web. The study indicates that online usage examples have already been a good resource for programmers to learn to use APIs. Due to the lack of comprehensive understanding on the current situation of usage examples on the web, utilization of these examples is still at the very beginning. More research effort should be invested to facilitate the utilization. From the perspective of example's users, how to help them find required examples and learn from the examples should be investigated. From the perspective of example's providers, we need to understand what can be done to help them produce and share good examples more easily. According to the study results, we draw some findings with the expectation that more techniques and researches being applied on this topic, and we look forward to seeing the research results.

ACKNOWLEDGMENT

This work is supported by the National Basic Research Program of China (973) under Grant No. 2009CB320703; the High-Tech Research and Development Program of China (863) under Grant No.2012AA011202; the National Natural Science Foundation of China under Grant No. 61103024, No. 60931160444.

REFERENCES

[1] S. Thummalapenta and T. Xie, "Parseweb: a programmer assistant for reusing open source code on the web," in Proceedings of the 22nd IEEE/ACM international conference on Automated software engineering, ASE '07. New York, USA, 2007, pp. 204-213.

[2] J. Kim, S. Lee, S. Hwang, and S. Kim, "Adding Examples into Java Documents", in Proceedings of the 24th IEEE/ACM international conference on Automated software engineering, ASE'09. Nov. 2009, pp. 540-544

[3] H. Zhong, T. Xie, *et al.*, "MAPO: Mining and recommending API usage patterns". in Proc. ECOOP 2009, pp. 318-343

[4] S.K. Bajracharya, J. Ossher, and C.V. Lopes, "Leveraging Usage Similarity for Effective Retrieval of Examples in Code Repositories", in Proc. FSE 2010, pp. 157-166.

[5] L. W. Mar, Y.-C. Wu, and H. C. Jiau, "Recommending proper API code examples for documentation purpose," in Proc. APSEC 2011, pp. 331-338.

[6] S. Thummalapenta and T. Xie, "SpotWeb: Detecting framework hotspots and coldspots via mining open source code on the web," in 2008 23rd IEEE/ACM International Conference on Automated Software Engineering. IEEE, Sep. 2008, pp. 327-336.

[7] Lijie Wang, Lu Fang, Leye Wang, Ge Li, Bing Xie, Fuqing Yang, "APIExample: An Effective Web Search Based Usage Example Recommendation System for Java APIs", 26th IEEE/ACM International Conference On Automated Software Engineering (ASE) 2011, pp.592-595

[8] Raphael Hoffmann, James Fogarty, and Daniel S. Weld, "Assieme: finding and leveraging implicit references in a web search interface for programmers", in Proc. UIST 2007, pp. 13-22

[9] Joel Brandt, Mira Dontcheva, Marcos Weskamp, et al. "Example-Centric Programming: Integrating Web Search into the Development Environment", in Proc. CHI 2010, pp. 513-522

[10] J. Stylos, B. A. Myers, "Mica: A Web-Search Tool for Finding API Components and Examples", in Proc. VL/HCC 2006, pp. 195-202

[11] C. Parnin and C. Treude, "Measuring API documentation on the web," in Proceeding of the 2nd international workshop on Web 2.0 for software engineering, ser. Web2SE '11. New York, NY, USA: ACM, 2011, pp. 25-30.

[12] C. Treude and M. A. Storey, "Effective communication of software development knowledge through community portals" in Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, ser. ESEC/FSE '11. New York, NY, USA: ACM, 2011, pp. 91-101.

[13] C. Treude, O. Barzilay, and M. A. Storey, "How do programmers ask and answer questions on the web? (NIER track)," in Proceeding of the 33rd International Conference on Software Engineering, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 804-807.

[14] O. Barzilay, A. Yehudai, and O. Hazzan, "Developers attentiveness to example usage," in Human Aspects of Software Engineering, ser. HAoSE '10. New York, NY, USA: ACM, 2010.

[15] S. E. Sim, M. Umarji, S. Ratanotayanon, and C. V. Lopes, "How well do search engines support code retrieval on the web?" ACM Trans. Softw. Eng. Methodol., vol. 21, no. 1, Dec. 2011.

[16] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne, "Finding high-quality content in social media," in Proceedings of the international conference on Web search and web data mining, ser. WSDM '08. New York, NY, USA: ACM, 2008, pp. 183-194

[17] A. J. Ko, B. A. Myers, and H. H. Aung, "Six learning barriers in End-User programming systems," in 2004 IEEE Symposium on Visual Languages - Human Centric Computing, ser. VLHCC '04, vol. 0. Los Alamitos, CA, USA: IEEE, 2004, pp. 199-206.

[18] R. E. Gallardo Valencia and S. E. Sim, "What kinds of development problems can be solved by searching the web?: a field study," in Proceeding of the 3rd international workshop on Search-driven development: users, infrastructure, tools, and evaluation, ser. SUITE '11. New York, NY, USA: ACM, 2011, pp. 41-44.

[19] D. Hou and L. Li, "Obstacles in using frameworks and APIs: An exploratory study of programmers' newsgroup discussions," in 2011 IEEE 19th International Conference on Program Comprehension. IEEE, Jun. 2011, pp. 91-100.

[20] M. P. Robillard and R. DeLine, "A field study of API learning obstacles," Empirical Software Engineering, vol. 16, no. 6, pp. 703-732, Dec. 2011.

[21] Brandt J, Guo PJ, Lewenstein J, Dontcheva M, Klemmer SR (2009) Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In Proc. 27th int'l conf. on human factors in computing systems, pp 1589–1598

[22] Nykaza J, Messinger R, Boehme F, Norman CL, Mace M, Gordon M (2002) What programmers really want: results of a needs assessment for SDK documentation. In: Proc. 20th annual ACM SIGDOC int'l conf. on computer documentation, pp 133–141

[23] C. Goues and W. Weimer, "Specification mining with few false positives," in Proceedings of the 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems: TACAS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 292-306.