Nuggets Miner: Assisting Developers by Harnessing the StackOverflow Crowd Knowledge and the GitHub Traceability

Eduardo C. Campos[1], **Lucas B. L. de Souza**[1], **Marcelo de A. Maia**[1]

[1]Department of Computer Science – Federal University of Uberlândia (UFU),
Uberlândia, MG, 38400-902, Brazil

{eduardocunha11,lucas.facom.ufu}@gmail.com, marcmaia@facom.ufu.br

Abstract. *StackOverflow.com (SOF) is a Question and Answer service oriented to support collaboration among developers. The information available on this type of service is also known as "crowd knowledge" and currently is one important trend in supporting activities related to software development. GitHub.com (GitHub) is a successful social site for developers that makes unique information about users and their activities visible within and across open source software projects. The traceability of GitHub's issue tracker can be harnessed in the Integrated Development Environment (IDE) to assist software maintenance. We give a form to our approach by implementing Nuggets Miner, an Eclipse plugin, that recommends a ranked and interactive list of results to system's user. Video Demo URL: https://www.youtube.com/watch?v=AjsbgUJl-nY*

1. Introduction

Modern-day software development is inseparable from the use of the Application Programming Interfaces (APIs) [Duala-Ekoko and Robillard 2012]. Several studies have shown that developers face problems when dealing with unfamiliar APIs [Duala-Ekoko and Robillard 2012, Holmes et al. 2006, Thung et al. 2013]. It is seldom the case that the documentation and examples provided with a large framework or library are sufficient for a developer to use their API effectively. Frequently, developers become lost when trying to use an API, unsure of how to make a progress on a programming task [Holmes et al. 2006]. A common behavior of developers is to post questions on social media services and receive answers from other programmers that belong to different projects [Treude et al. 2011].

To help developers find their way, a widely-know alternative is StackOverflow (SOF), which is a Question and Answer (Q&A) website which uses social media to facilitate knowledge exchange between programmers by mitigating the pitfalls involved in using code from the Internet. Mamykina et al. conducted a statistical study of the entire SOF corpus to find out what is behind the immediate success of it. Their findings showed that a majority of the questions will receive one or more answers (above 90% very quickly - with a median answer time of 11 minutes) [Mamykina et al. 2011]. The set of information available on this social media services is called "crowd knowledge" and often become a substitute for the official software documentation [Treude et al. 2011].

Despite its usefulness, the knowledge provided by Q&A services cannot be directly leveraged from within an Integrated Development Environment (IDE), in the sense that developers must toggle to the Web browser to access those services. Moreover, when dealing with maintenance tasks, software developers often also need to know what changes were made in the past of the project. Thus, the developers are forced to explore the historical information of the project (e.g., issues and respective commits) [Robillard and Dagenais 2010]. In order to address this problem, we examined a successful social site called GitHub [1]. This site makes unique information about users and their activities visible within and across open source software projects [Dabbish et al. 2012]. Furthermore, the GitHub's issue tracker has excellent traceability and this feature can be harnessed in the IDE (e.g., given a closed issue, we can explore the respective commit). Although GitHub's site has an integrated issue tracker, it is not possible to search automatically for related issues to a particular maintenance task. Thus, during the maintenance task, the developer is constantly reviewing the issue tracker in search for some issue related to your task. Concluding, developers spend most of their time in the IDE to write and understand code [LaToza et al. 2006] and they should be only focused on the current task without any major interruption or disturbance [Raskin 2000]. Nevertheless, developers are forced to leave the IDE, thus interrupting the programming flow and lowering their focus on the current task, and also maybe getting distracted with other activities in the Web.

To deal with those problems, recommendation systems can be a reasonable alternative option. According to Robillard et al., a recommendation system for software engineering (RSSE) can assist developers during maintenance and development tasks providing useful information (e.g., right code for a task, good example of API usage) [Robillard et al. 2010]. This information can be gathered from the "crowd knowledge" provided by Q&A services or gathered from closed issues in GitHub related to the current maintenance task.

Considering StackOverflow, we can rely on regular dumps of the entire dataset to obtain the desired information. In the case of GitHub, the current project that the developer is working on must host their issues in the GitHub's issue tracker instead of other issue trackers (e.g., Bugzilla [2]). Nuggets Miner extracts only issues with CLOSED state (i.e., issues that were previously solved by other developers), displays the ranked search issues directly in the IDE and allows developers to select an issue and explore the historical changes made to the respective commit files.

Our work has the following contribution. We present Nuggets Miner, a recommendation system in the form of a plugin for Eclipse IDE [3] to assist software developers in development and maintenance tasks. Our recommendation strategy has been partially assessed in [Souza et al. 2014] (i.e., only recommendations of SOF were evaluated). There are several differences from this paper and [Souza et al. 2014] paper, but these two are the most important: 1) that paper was not tool-oriented, indeed, no tool was presented; 2) that paper was only about SOF posts, but Nuggets Miner also indexes project's issues.

The rest of this paper is organized as follows. In Section 2 we illustrate Nuggets Miner usage with a use case scenario. In Section 3 we present Nuggets Miner components and its architecture. In Section 4 we discuss related work. In Section 5 we draw our

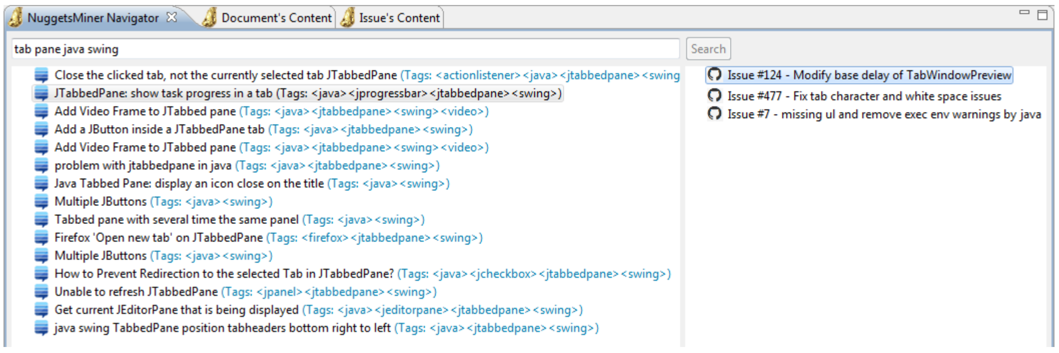
[1]https://github.com/
[2]http://www.bugzilla.org/
[3]http://eclipse.org

conclusions.

2. A Use Case Scenario

We show how Nuggets Miner can help developers to solve programming problems by leveraging SOF and GitHub traceability from within the Eclipse IDE.

Bob is required to build a panel with three tabs using Java Swing API. However, he is novice in this library. Bob opens up the Eclipse IDE, with the Nuggets Miner plugin installed and writes the following query in Nuggets Miner's Navigator: "tab pane java swing". Figure 1 shows the search results returned by the search engine for the query "tab pane java swing": Q&A pairs in the left panel and issues in the right panel. Concerning the StackOverflow panel, the search engine returns to Bob, the top 15 Q&A pairs from SOF in a ranked list considering two main aspects: the textual similarity of the pairs with respect to the query and the quality of pairs (whose content was previously evaluated by SOF community). Among the recommended Q&A pairs, Bob finds out a pair whose title is "JTabbedPane: show task progress in a tab". Figure 2 shows the content of a selected Q&A pair. He reads the Q&A pair and finds an accepted answer that creates an object of *JTabbedPane* type and invokes the method *public void addTab(String title, Icon icon, Component component)* of this object. Bob can also import the code snippet given in the answer into program's editor via drag & drop and execute the Java program (in this case without any modification). Thus, Bob can start modifying the code in the editor to achieve the desired outcome. Concerning the Github panel, in the list of returned issues, Bob can choose an issue that he thinks is more related to his activity. Figure 3 shows the content of a selected issue. He can visualize the conversations between Bob's colleagues about the selected issue (through Conversation tab) which is supposed to be related to his current task. In the Figure 3, it is possible to visualize the list of commits with their respective links (through Commits tab). When Bob clicks in the commit's link, another page will open showing the code modified by the commit. Figure 4 shows a snapshot of commit selected by Bob.



tab pane java swing

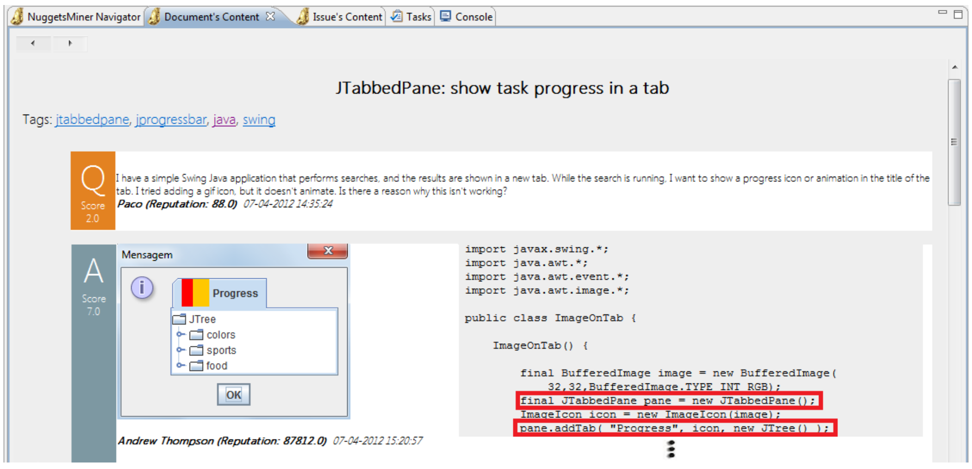
Close the clicked tab, not the currently selected tab JTabbedPane (Tags: <actionlistener><java><jtabbedpane><swing
JTabbedPane: show task progress in a tab (Tags: <java><jprogressbar><jtabbedpane><swing>)
Add Video Frame to JTabbed pane (Tags: <java><jtabbedpane><swing><video>)
Add a JButton inside a JTabbedPane tab (Tags: <java><jtabbedpane><swing>)
Add Video Frame to JTabbed pane (Tags: <java><jtabbedpane><swing><video>)
problem with jtabbedpane in java (Tags: <java><jtabbedpane><swing>)
Java Tabbed Pane: display an icon close on the title (Tags: <java><swing>)
Multiple JButtons (Tags: <java><swing>)
Tabbed pane with several time the same panel (Tags: <java><swing>)
Firefox 'Open new tab' on JTabbedPane (Tags: <firefox><jtabbedpane><swing>)
Multiple JButtons (Tags: <java><swing>)
How to Prevent Redirection to the selected Tab in JTabbedPane? (Tags: <java><jcheckbox><jtabbedpane><swing>)
Unable to refresh JTabbedPane (Tags: <jpanel><jtabbedpane><swing>)
Get current JEditorPane that is being displayed (Tags: <java><jeditorpane><jtabbedpane><swing>)
java swing TabbedPane position tabheaders bottom right to left (Tags: <java><jtabbedpane><swing>)

Issue #124 - Modify base delay of TabWindowPreview
Issue #477 - Fix tab character and white space issues
Issue #7 - missing ul and remove exec env warnings by java

Figure 1. Nuggets Miner's Navigator: Search Results.

3. Nuggets Miner

In this section, we present Nuggets Miner's architecture (Subsection 3.1), the mechanism for data collection and classification (Subsection 3.2) and the query engine (Subsection 3.3).



JTabbedPane: show task progress in a tab

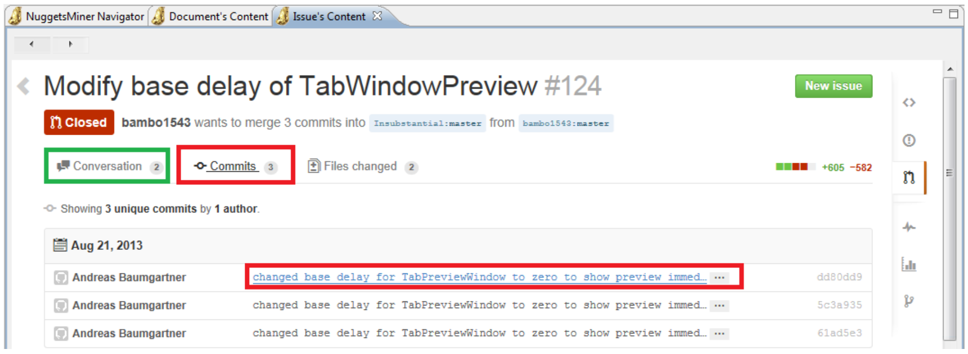
Tags: jtabbedpane, jprogressbar, java, swing

I have a simple Swing Java application that performs searches, and the results are shown in a new tab. While the search is running, I want to show a progress icon or animation in the title of the tab. I tried adding a gif icon, but it doesn't animate. Is there a reason why this isn't working?
Paco (Reputation: 88.0) *07-04-2012 14:35:24*

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;

public class ImageOnTab {

    ImageOnTab() {

        final BufferedImage image = new BufferedImage(
            32,32,BufferedImage.TYPE_INT_RGB);
        final JTabbedPane pane = new JTabbedPane();
        ImageIcon icon = new ImageIcon(image);
        pane.addTab( "Progress", icon, new JTree() );
```

Andrew Thompson (Reputation: 87812.0) *07-04-2012 15:20:57*

Figure 2. Nuggets Miner’s Document’s Content for the Q&A pair selected by Bob.



Modify base delay of TabWindowPreview #124

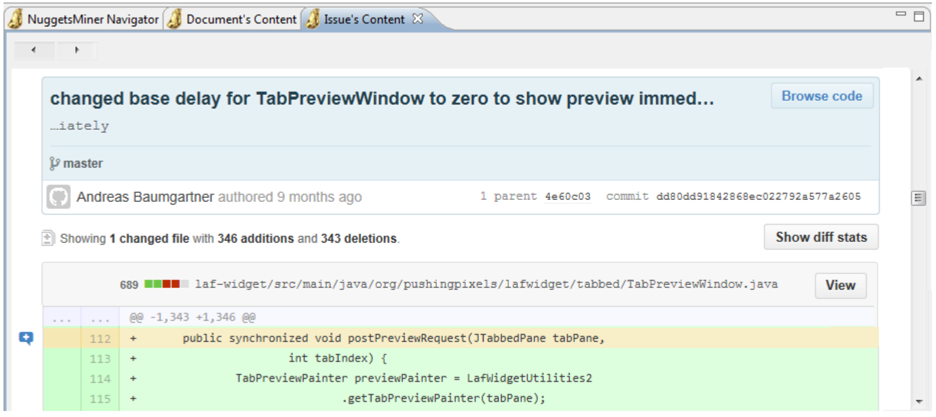
bambo1543 wants to merge 3 commits into `Insubstantial:master` from `bambo1543:master`

Showing **3 unique commits** by **1 author**.

Aug 21, 2013

Andreas Baumgartner	changed base delay for TabPreviewWindow to zero to show preview immed...	dd80dd9
Andreas Baumgartner	changed base delay for TabPreviewWindow to zero to show preview immed...	5c3a935
Andreas Baumgartner	changed base delay for TabPreviewWindow to zero to show preview immed...	61ad5e3

Figure 3. Nuggets Miner’s Issue’s Content for the issue selected by Bob.



changed base delay for TabPreviewWindow to zero to show preview immed...

...iately

Andreas Baumgartner authored 9 months ago   1 parent 4e60c03   commit dd80dd91842868ec022792a577a2605

Showing **1 changed file** with **346 additions** and **343 deletions**.

689 laf-widget/src/main/java/org/pushingpixels/lafwidget/tabbed/TabPreviewWindow.java

```
@@ -1,343 +1,346 @@
+       public synchronized void postPreviewRequest(JTabbedPane tabPane,
+                       int tabIndex) {
+               TabPreviewPainter previewPainter = LafWidgetUtilities2
+                               .getTabPreviewPainter(tabPane);
```

Figure 4. Snapshot of commit selected by Bob.

3.1. The Architecture

Figure 5 depicts Nuggets Miner’s architecture. The left part of this figure represents the server side, while the right side represents the client side (i.e., the graphical user interface and features of the plugin).

On the server side, there is a component for collection and classification of data, which is responsible for collecting and classifying Q&A pairs from SOF. Through this component, we can also collect issues with CLOSED state along with the respective commits for a given interest project hosted on GitHub. Therefore, the Apache Solr [4] index is constructed with Q&A pairs from SOF and with issues and respective commits of the developer's project hosted in the GitHub.

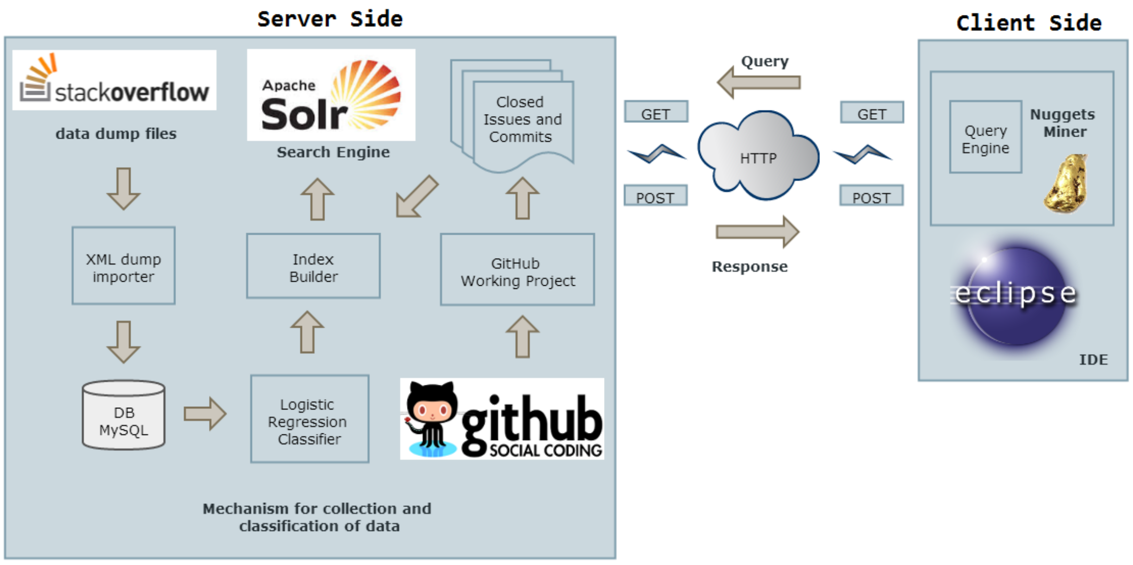


Figure 5. Nuggets Miner's architecture.

The client side is responsible for querying the Apache Solr index, parsing the JSON response (converting the JSON in a object-oriented representation for further manipulation), applying the methodology for ranking the Q&A pairs, applying the methodology for ranking the related issues and present these search results to the user's system.

The ranking criteria for Q&A pairs is based on two main aspects: the textual similarity of the pairs with respect to the query and the quality of the pairs (assessed by SOF community members), while the ranking for GitHub issues takes into account only the textual similarity of the issues with respect to the query.

3.2. Mechanism for Data Collection and Classification

In this subsection, we explain the mechanism for data collection (Subsection 3.2.1) and the mechanism for data classification for Q&A pairs (Subsection 3.2.2).

3.2.1. Data Collection

We downloaded a release of SOF public data dump [5] provided by Stack Exchange [6], which comprises several XML files that represent the database of each website. Since performing these operations by manipulating data directly from XML files is resource intensive, we imported everything in a relational database in order to classify the SOF Q&A pairs. The "posts" table of this database stored all questions posted by questioners

[4]http://lucene.apache.org/solr/
[5]http://blog.stackexchange.com/category/cc-wiki-dump/
[6]http://stackexchange.com/

in the website until the date the dump was performed. This table also stores all answers that were given to each question, if any.

To retrieve the issues with CLOSED state from a GitHub project, we developed another program that connects in the GitHub server (informing the user and password) and performs the download of these issues from a given repository (e.g., in our study we considered a Swing look-and-feel project called Insubstantial [7] that is hosted on GitHub[8]). Our program used an object-oriented GitHub API [9]. Then, for each retrieved issue, the program stores the issue data (e.g., "issue title", "issue body", "issue id", "commit address of the issue in GitHub", "code modified by the commit") in a XML file in the format required by Apache Solr search engine. For instance, the issue whose "id" is #124 belongs to the repository *"Insubstantial/insubstantial"*. The "issue title" of this issue is: *"Modify base delay of TabWindowPreview"*. The commit address of this issue in GitHub is: "*https://github.com/Insubstantial/insubstantial/pull/124/commits*". This page will be displayed inside the browser of Nuggets Miner plugin.

3.2.2. Data Classification for Q&A pairs

On SOF, users ask many kind of different questions. Accordingly to Nasehi et al. [Nasehi et al. 2012] "questions from SOF can also be classified in a second dimension that is about the main concerns of the questioners and what they wanted to solve". In this dimension, one of the categories is the *How-to-do-it* in which the questioner provides a scenario and asks about how to implement it. This category is very close to scenario in which a developer has a programming task at hand and needs to solve it. For this reason, in our approach, we only consider Q&A pairs that are classified as *How-to-do-it*. In order to automate the selection of this kind of Q&A pairs, we developed a classification strategy. The information about the categories of Q&A pairs proposed in this study, the classifier's attributes and the steps to build the dataset for training/test of the classifier are described in more detail in [Souza et al. 2014].

We used this classifier to automatically classify Q&A pairs of a pre-selected set of APIs (Swing of Java, Boost of C++ and LINQ of C#) into one of three categories: *How-to-do-it*, *Conceptual* and *Seeking-something*. The Apache Solr index was populated with only Q&A pairs of *How-to-do-it* category of these pre-selected set of APIs.

3.3. The Query Engine

Nuggets Miner's Eclipse plugin makes the Q&A "crowd knowledge" and closed issues of a working GitHub project available in the IDE. Users can interact with this "crowd knowledge" in ways that the SOF website normally does not allow, such as import code snippets to the program's editor through simple drag & drop. The main goal of the query engine is to communicate with Apache Solr, by creating a query given an input string. It is necessary that the Q&A pair has some information on your "title", "question body" or "answer body" to be returned by the search engine. It is also needed that the GitHub issues has some information on your "issue title", "issue body" or "code modified by

[7]http://shemnon.com/speling/2011/04/insubstantial-62-release.html

[8]https://github.com/Insubstantial/insubstantial

[9]http://github-api.kohsuke.org/

the commit" to be returned by the search engine. As stated above, the query engine simultaneously queries Apache Solr index for both Q&A pairs and issues similar to the entered query. The query engine tokenizes the string inserted by the developer. The engine builds the query, according to Apache Solr syntax, in a way that every token must be presented in the document fields.

4. Related Work

Ponzanelli et al. [Ponzanelli et al. 2013] presented an approach to assist programmers who want to leverage the "crowd knowledge" of Q&A services. They implemented SEAHAWK, a recommendation system in the form of a plugin for the Eclipse IDE to harness the "crowd knowledge" of SOF from within the IDE. In our work, we introduced a more efficient ranking mechanism than SEAHAWK and provided the GitHub access point. We used SEAHAWK software[10] to help us developing our plugin.

Cordeiro et al. [Cordeiro et al. 2012] presented an Eclipse plugin to help developers in problem solving tasks. Based on an exception's stack trace gathered from the IDE's console, they suggest related documents from SOF.

HIPKAT [ČubraniĆ et al. 2004] is a recommendation system developed to support newcomers in a project by recommending items from problem reports, newsgroup, and articles. Our approach recommends project's issues with CLOSED state related to the maintenance task at hand.

Takuya et al. presented SELENE [Takuya and Masuhara 2011], a source code recommendation tool based on an associative search engine. It spontaneously searches and displays example programs while the developer is editing a program text. Our work also relies on search engines, but we suggest Q&A pairs taken from SOF to enrich the information provided by code snippets.

5. Conclusions

We presented a novel approach to leverage the SOF "crowd knowledge" and the GitHub traceability. We have detailed the implementation of our approach, Nuggets Miner. This recommendation system allows users interact with SOF knowledge by importing code snippets. It also allows users navigate through related issues previously solved by others developers. Thus, users can explore the respective commit for the recommended issue and see the modifications. As future work, we intend to improve the evaluation of Nuggets Miner with human subjects to assess the performance gains compared to the use of an external browser.

6. Acknowledgments

This work was partially supported by FAPEMIG grant CEXAPQ-2086-11 and CNPQ grant 475519/2012-4.

References

Cordeiro, J., Antunes, B., and Gomes, P. (2012). Context-based recommendation to support problem solving in sof. development. In *Proceedings of 3rd Int. Workshop on RSSE)*, pages 85–89.

[10]http://seahawk.inf.usi.ch/download.html

Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. (2012). Social Coding in Github: Transparency and collaboration in an open software repository. CSCW '12, pages 1277–1286. ACM.

Duala-Ekoko, E. and Robillard, M. P. (2012). Asking and answering questions about unfamiliar apis: An exploratory study. In *Proc. of ICSE'2012*, pages 266–276. IEEE Press.

Holmes, R., Walker, R. J., and Murphy, G. C. (2006). Approximate structural context matching: An approach to recommend relevant examples. *IEEE Trans. Softw. Eng.*, 32(12):952–970.

LaToza, T. D., Venolia, G., and DeLine, R. (2006). Maintaining mental models: A study of developer work habits. In *Proc. of ICSE'2006*, pages 492–501. ACM.

Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G., and Hartmann, B. (2011). Design lessons from the fastest q&a site in the west. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2857–2866. ACM.

Nasehi, S., Sillito, J., Maurer, F., and Burns, C. (2012). What makes a good code example? A study of programming Q&A in Stack Overflow. In *Proc. of ICSM'2012*, pages 25–34.

Ponzanelli, L., Bacchelli, A., and Lanza, M. (2013). Leveraging crowd knowledge for software comprehension and development. In Cleve, A., Ricca, F., and Cerioli, M., editors, *Proc. of CSMR'2013*, pages 57–66. IEEE Computer Society.

Raskin, J. (2000). *The Humane Interface: New Directions for Designing Interactive Systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

Robillard, M. P. and Dagenais, B. (2010). Recommending change clusters to support software investigation: An empirical study. *J. Softw. Maint. Evol.*, 22(3):143–164.

Robillard, M. P., Walker, R. J., and Zimmermann, T. (2010). Recommendation systems for software engineering. *IEEE Software*, 27(4):80–86.

Souza, L., Campos, E., and Maia, M. (2014). Ranking crowd knowledge to assist software development. In *Proc. of ICPC'2014*, pages 1–11.

Takuya, W. and Masuhara, H. (2011). A spontaneous code recommendation tool based on associative search. In *Proceedings of the 3rd International Workshop on Search-Driven Development*, pages 17–20. ACM.

Thung, F., Wang, S., Lo, D., and Lawall, J. L. (2013). Automatic recommendation of api methods from feature requests. In *ASE*, pages 290–300. IEEE.

Treude, C., Barzilay, O., and Storey, M.-A. (2011). How do programmers ask and answer questions on the web? (nier track). In *Proc. of ICSE'2011*, pages 804–807. ACM.

ČubraniĆ, D., Murphy, G. C., Singer, J., and Booth, K. S. (2004). Learning from project history: A case study for software development. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 82–91. ACM.