Enhancing intelligent API tutors with information about how APIs are used in practice
AN ABSTRACT OF THE PROJECT OF

Vasanth Krishnamoorthy for the degree of Master of Science in Computer Science presented on June 14, 2012.

Title: Enhancing intelligent API tutors with information about how APIs are used in practice

Abstract approved: ______________________________________________________

Christopher Scaffidi

Application Programming Interfaces (APIs) enable software developers to utilize and create functionality that would otherwise take a lot of time and effort to build from scratch. Consequently, an essential part of software engineering training is for students to learn how to use APIs effectively. The existing jTutors system enables an expert library user to harvest code examples from the web and semi automatically create intelligent tutors for various Java APIs. This interactive instructional material teaches a learner how to perform tasks with APIs through a series of examples and quizzes presented in a computer adaptive manner. This thesis presents two new features that enhance the jTutors system by incorporating additional information into intelligent tutors about how APIs are used in practice. The new 'Industry Code Search' feature enables users to search for API code usages by mining through a repository of open source projects. This system has been further enhanced to mine FAQs, articles and other API related links so users can learn more about APIs, all in one place. A qualitative study demonstrated that the new features helped students to learn intricacies of multiple APIs and also to familiarize themselves with actual usage in real world projects.
ACKNOWLEDGEMENTS

First and foremost, my utmost gratitude to my adviser Dr. Christopher Scaffidi for his motivation, enthusiasm, immense knowledge and continuous support of my MS study and research. His guidance helped me in all times during my research and thesis. I couldn’t have imagined having a better adviser and mentor for my Masters study.

To Dr. Margaret Burnett and Dr. Carlos Jensen, my committee members, I am very much grateful for the knowledge and insights you have imparted for the improvement of my work. The courses you taught have helped me a lot in shaping myself as a better software engineer.

Oregon State University provided me an excellent atmosphere for studying and research. I’d also like to extend a special thanks to the EECS staff for giving me an opportunity to work as a Web Developer in the department and funding me as well.

I thank my fellow researchers Aniket Dahotre, Matt Corley and Bharatwaj Appasamy for the team work and contributions with implementing jTutors. Thanks to Aravind, Balaji, Barath, Deepthi, Hariprasath, Harsha, Kunal, Nishanthini, Nitin, Pradyumna, Pratheeksha, Roma, Sanchit, Santosh, Satish, Sekhar, Shalini, Sharath, Vignesh, Vivek, ISA committee members and many others for being such wonderful friends. Special thanks to Akhil for being such a great friend, roommate and the delicious food he cooked for us.

Last but not the least; I would like to thank my parents Krishnamoorthy, Manivasuki and brother, Swarun Krishna for being supportive of me throughout my life. Above all, I humbly thank God Almighty, whose grace and blessing has been guiding me in the right path all through my life.
# TABLE OF CONTENTS

1 Introduction ........................................................................................................................................ 1

2 Related Work and Background ......................................................................................................... 3

2.1 Challenges of learning APIs ........................................................................................................ 3

2.2 Existing jTutors system ................................................................................................................ 5

2.3 Strengths and Weaknesses of current tutoring systems ................................................................. 11

3 Approach ........................................................................................................................................ 12

3.1 Integrating FAQs and other supplementary pages .................................................................... 13

3.2 Integrating repository code ........................................................................................................ 16

3.3 Cognitive dimensions in design decisions .................................................................................. 21

3.4 Implementation Details ............................................................................................................. 24

4 Evaluation ....................................................................................................................................... 31

4.1 Study Methodology .................................................................................................................... 32

4.2 Analysis Methodology and Results ............................................................................................ 37

5 Conclusion and Future Work ........................................................................................................... 46

6 Bibliography .................................................................................................................................... 50
LIST OF FIGURES

Figure 1. Searching for API topic. ................................................................. 6
Figure 2. Tutorial selection page ............................................................... 7
Figure 3. Compiler messages are shown if any error .................................. 8
Figure 4. The description of the API is shown before launching the tutor ...... 9
Figure 5. Shows a jTutors example with API related words underlined ....... 10
Figure 6. A quiz screen with the green box indicating a correct answer .......... 11
Figure 7. Creating intelligent tutors using jTutors .................................... 13
Figure 8. Shows the API related links returned to the teacher. ................. 15
Figure 9. Users can also rate each API related link .................................... 16
Figure 10. Work flow of industry Code Search feature. .......................... 17
Figure 11. Teachers can upload multiple files to the system, using the browse button. .......... 17
Figure 12. Student can search for API method usages here ....................... 18
Figure 13. Search results page for Industry Code search feature. ............... 20
Figure 14. Sample code returned for Industry Code Search ....................... 21
Figure 15. Result preview for the ‘Industry Code search’ feature .................. 22
Figure 16. The FAQs, tutorials, Articles and Related links are juxtaposed next to each other. ... 23
Figure 17. Lucene Query Code: The “String q” in the code indicates the query. .......... 24
Figure 18. Teacher is presented with API related links ............................... 27
Figure 19. Luke is eased to for viewing and modifying index files generated .................................. 30
Figure 20. Pie chart shows the distribution of features that users felt was very useful 39
Enhancing intelligent API tutors with information about how APIs are used in practice

1 Introduction

Java is one of the most well-known programming languages used to create Web applications and mobile platforms. Java currently runs in more than 850 million computers worldwide and on billions of other devices like mobile and TV [1]. With the growing importance of Java as a programming language it has become increasingly important to learn to use it effectively by building upon already existing components to create powerful applications.

In order to create or extend software it is normal to use pieces of code that are already available and reusable in the form of libraries. A study indicates that programmers spend 20 out of every 100 hours learning how to use libraries, which is far longer time than they spend actually writing code [2]. Programmers find it difficult learning to use libraries because they mainly rely on code examples to support learning which is quite difficult to find on the web [3][4][5][6]. Consequently, an essential part of software engineering training for students is to teach them to effectively use APIs.

An ACM (Association for Computing Machinery) task force on pedagogy explains, “The most serious problem facing instructors who try to teach Java—or any modern industrial-strength language for that matter—is the problem of scale” [1]. Especially, APIs are vast, typically having thousands of functions [1]. Traditional instruction techniques, such as lectures, can only expose students to a small fraction of these API functions, so they must instead learn specialized APIs while actually working with it in a less productive and time consuming manner [8][3][4]. Moreover, code built using an API can in turn become a new API, and thus companies
often create their own proprietary APIs for their use [12]. This presents a periodic need to keep learning new APIs even for professional programmers.

Novice programmers usually learn APIs by looking into online tutorials while they are struggling with API concepts [3] [4]. This approach is favored in most cases where API documentation is difficult to understand or missing [5]. An alternative solution is to use intelligent tutors, which are instructional materials that teach students to perform tasks and tailor itself based on the student’s progress [26][27]. A prior study to evaluate our system (jTutors) showed that the quizzes and examples in intelligent tutors helped in reducing the learning time of APIs by 33% when compared to textbooks [11]. Based on feedback from this study, we found that people wanted features that could show commonly made mistakes and other important links related to each API being taught. Participants also suggested having a search feature for viewing API related code samples from actual real world projects. Thus, there is a need for users to familiarize themselves with information about how APIs are used in actual projects to implement different functionality.

In view of this problem, we implemented new features that enhance our system by providing a strategy for users to learn the intricacies of multiple APIs. The 'Industry Code Search' feature enables users to search for API code usages by mining through a repository of open source projects. The system also mines FAQs, articles and other API related links so users can learn more about APIs, all in one place. A qualitative study conducted in a laboratory setting demonstrated the effectiveness of these new features in helping students learn APIs.
2 Related Work and Background

An API (Application Programming Interface) is a set of methods and classes implemented as a code library that can be reused in multiple instances across multiple applications. For example, Java’s standard library currently has more than 34,000 methods and classes and Microsoft .NET library has more than 140,000 [15]. This number is continuously increasing and popular repositories have around 160,000 available for download [16]. In addition to this, companies individually produce their own proprietary libraries for private use [17]. These libraries enable their programmers to re-use code via their APIs. Today, writing useful programs is practically impossible without reuse and these libraries help in serving as building blocks using which functionality is built [1]. It is understood from numerous empirical studies of actual industry code reuse libraries (called “blackbox reuse”) that API usage is one of the “single most effective” methods ever used to boost programmer’s productivity [19] [20].

2.1 Challenges of learning APIs

The presence of such a vast number of software libraries presents a need for programmers to learn to use them. This challenge is faced by programmers both novice programmers in the classroom context as well as in the professional context, since creating or extending software continually demands programmers to use new libraries which they need to master. One study shows that professional programmers spent 20% of their day learning to use APIs [2] and only 12% of the time is actually spent writing code. In another study, when programmers were asked why learning libraries took so long, 60% of the respondents found training resources to be inadequate, mainly due to “insufficient or inadequate examples” [5].
Students in computer science often struggle with learning to use APIs, especially in courses that have team based projects where each member codes a part of the software [21][22][23]. These students are already well versed with programming constructs and primitives, but still need to figure out how to use their code to combine different APIs and create useful software. Helping students with these projects can be very difficult mainly because each project has different requirements and calls for the use of different libraries [24]. Each team uses different libraries (eg. for displaying multimedia, controlling mobile phones, voice-response systems or even to implement drivers [24][25]). It’s almost impossible for a student to have known all of these libraries—nor is there any need to do so, since most libraries are outside the area of interest of the student. Similarly, it’s quite impossible for an instructor to conduct a course hoping to teach the full range of libraries required for all of the projects. Thus there is a need that each student must learn specialized APIs with some kind of minimal instructional support. This issue were the number of libraries actually surpasses the knowledge of the instructor was identified as “the most serious problem facing instructors who try to teach Java” by a task force of the Association for Computing Machinery (the world’s largest academic computing society) [1].

One of the approaches potentially useful for solving this problem is to use intelligent tutors, which provides customized instruction or feedback based on the student’s learning progress. The Cognitive Tutor Authoring Tools (CTAT) suite is one of the most well-known systems for creating intelligent tutors [28]. With CTAT, a teacher can create tutorial screens by simply dragging and dropping labels or textboxes into the appropriate places. For a quiz task,
the teacher demonstrates valid and invalid solution methods and from these examples, CTAT automatically generates a generalized example.

CTAT has a series of examples and quizzes combined together into a Java or Flash program that constitutes the intelligent tutor and can then be used by students. The student’s progress in the tutorials is reported back to a website and additional tutors are recommended to the student based on assessment of his strengths and weaknesses. Studies show that intelligent tutors increase learning in a given amount of time when compared to that of conventional textbook learning [29][30]. They have proven to be very useful in teaching fundamental programming concepts such as loops and records [26][31][32] [33] [34].

### 2.2 Existing jTutors system

Our approach is for instructors to delegate some training responsibilities to jTutors, which harvests code examples from the web and package them into a series of quizzes/examples in the form of an intelligent tutor. The system tailors itself to the learner’s progress and computer adaptively recommends the next example or quiz. The process of creating examples/quizzes for APIs using jTutors is explained with an example in the following paragraphs.

The teacher decides to create a tutor for “Connecting to Database” and help the students with learning API methods related to it. The teacher now types “Connecting to Database” and clicks the Search button in the ‘Create Tutors’ tab (Figure 1).
jTutors uses popular third party search engines to find code samples from the World Wide Web. We currently use Google’s Custom search API and Bing API for retrieving results related to the query. The top 30 results obtained from both the search engines are screen scraped to retrieve java code samples. The code thus obtained is parsed, validated and passed through a series of filters to obtain relevant code snippets.

jTutors presents the code snippets to the teacher based on the order of relevance. It also identifies words in the snippets that demonstrate usage related to the current API query. These words can be used as suggestions for API methods that could be blanked out in fill-up type
quizzes. The code snippets returned by the system are verified by the teacher to see if they can be used as a quiz/example for the current API and if not they are discarded. This tutor selection screenshot is shown below in figure 2.

![Tutorial selection page]

Figure 2. Tutorial selection page

The teacher can also access compiler messages for the code snippets, by clicking “Toggle compiler messages”, found at the bottom of the page (as shown in figure 3). If the errors are minor, the teacher can go on to customize the code in the snippet by selecting the “edit” option and proceed to modify code. In case the teacher feels that the snippet has too many errors to be used in the tutor she could select the “discard” option for it.
While creating a quiz, the teacher can blank methods by using the list of API methods recommended. Hints can also be added to help the student answer each of the API methods in the blanks selected. These hints would be shown to the student during the actual tutorial by clicking on the “Get Hints” button. The teacher can also blank out methods not recommended by jTutors, by adding methods from the snippet to the quiz. Once the teacher has finished choosing relevant quizzes and example snippets, she can save them together by adding a small description about what the API is used for and it’s most common methods. The tutor is now packaged with quizzes/examples and available to access for all students.

Figure 3. Compiler messages are shown if any error
The student can login into their student account to access saved tutorials by searching API topics in the ‘Find tutors’ tab. They can also view the most popular tutors and also rate tutorials which they have used. Clicking on any of the tutorial topics shows the API’s description and below it is the “Launch jTutors” button which when clicked starts the intelligent tutors (shown in figure 4).

![jTutors](image)

**Figure 4.** The description of the API is shown before launching the tutor.
In the start of any API tutor, an example code of the current topic is shown to the user (figure 5). The learner pays attention to the underlined words and then clicks on “Done”. Next, a quiz with least difficulty level is presented with blanks to fill. The ‘Hints’ button can be used to check hints for the methods to be entered in the blanks. The first two hints provide clues to fill in the blank and the third hint provides the answer by itself. Entering a wrong answer in the blank turns the textbox to red and a right answer shows up in green.

**Connecting to a database using Java Database Connectivity (JDBC)**

API: javadatabaseconnectivity

```
try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch (Exception e) {
        System.out.println("Failed to load JDBC/ODBC driver.");
        return;
    }
try {
    ocn = DriverManager.getConnection("jdbc:odbc:mydb","parst","mojava");
    } catch (Exception e) {
        System.err.println("problems connecting to "+URL);
    }
```

Figure 5. Shows a jTutors example with API related words underlined.

During the quiz, the student’s actions are logged and scores are calculated to find out if the user knows the API methods well or not. Using more than one hint is an indication that the user might still need to learn more about the API and hence next up he is either presented with an example or a quiz with lesser difficulty level. If the student enters the right answers without many hints or incorrect attempts, then he/she progresses up the learning bar and the next quiz presented is of higher difficulty level. Thus, the sequence of quizzes and examples are
presented in a computer adaptive manner tailoring itself to the user’s learning capabilities. A sample quiz is shown in figure 6 below.

**Connecting to a database using Java Database Connectivity (JDBC)**

```java
statement stmt = conn.createStatement();
try {
    stmt.executeUpdate("INSERT INTO MyTable( name ) VALUES ( 'my name' )");
} finally {
    try {
        stmt.close();
    } catch (Throwable ignore) {
    }
}
```

Figure 6. A quiz screen with the green box indicating a correct answer.

### 2.3 Strengths and Weaknesses of current tutoring systems

Intelligent Tutoring Systems (ITS) concentrate on providing learners with instructions and help while learning new topics [35]. Such systems track the user’s learning progress and provide assistance to improve on their weak areas. Artificial intelligence and machine learning techniques are used to help systems build an understanding of the learner’s knowledge level [36]. The research contribution through jTutors provided a novel way of creating intelligent tutorials by leveraging code snippets available on the Internet [11]. It also created a platform for sharing tutorials among the Java teaching community.

A laboratory experiment conducted earlier showed that jTutors helped in reducing the learning time for libraries by 33% when compared to textbooks. However, feedback from participants suggested the need for improving the system to help learn more about APIs apart
from just method names and usages. Showing FAQs and code samples from actual projects could be particularly useful during code reviews, for answering conceptual questions and to understand unexpected behavior of APIs. Specifically, jTutors shows code that explains concepts, but not really how APIs are used in real world projects. This was considered to be the biggest weakness of jTutors. This motivated us to add new features to help address this issue. The approach and implementation details of these features are discussed in the following sections.

3 Approach

jTutors analyzes the structure of the code obtained from different sources and presents them in the form of examples and quizzes in the tutorial section. New features have now been implemented to check FAQs and other API related web links. These links are scraped from well-known programming sites and 3rd party search results. The teacher reviews all the web links suggested by jTutors and categorizes it into FAQs, tutorials, articles and related links. The choices are saved in the database and then presented to the learner in the browser when they use the system to learn APIs.

An “Industry code search” has also been implemented, where the user can search for API method usages in real world software code uploaded to jTutors by the teacher. This helps the learners to familiarize themselves with different API method usages in actual projects. This feature is convenient to allow users to find how different APIs are used together to implement common functionality. The approach and work flow of jTutors (shown in the figure 7) is explained in detail in the following sections.
3.1 Integrating FAQs and other supplementary pages

Tutorials in the form of quizzes/examples can be created using our previous jTutors system for various API topics. However in most cases, learning only about API method names might not help to understand the intricacies of an API in actual usage. Thus, jTutors has been upgraded to scrape FAQs, tutorials and other API related articles from popular websites and 3rd party search engines.
3.1.1 Scraping API links from popular programming websites

Based on research from various online sources and supporting publications, we identified
5 popular sites were programmers frequently visited to learn about API concepts [40]. These
websites have a strong Java user base which makes it ideal for our purpose of retrieving
information about frequently asked questions. In cases where the websites don’t return useful
results, we use 3rd part search engines to scrape API related links. The websites used are listed
below,

- Stackoverflow (http://stackoverflow.com/)
  - Most popular language-independent, collaboratively edited question and
    answer site for programmers
- DZone (http://www.dzone.com/)
  - Online tech publishing company with valuable content for software
    architects and developers worldwide.
- Java2s (http://www.java2s.com/)
  - Java2s.com emphasizes on examples over explanation and practice over
    theory.
- jGuru (http://www.jguru.com/)
  - A customizable portal with online training, FAQs, regular news updates,
    and tutorials on Java topics.
- JavaBlogs (http://javablogs.com)
  - JavaBlogs is a blog community that is a group aggregation site for blogs
    which discuss Java technology regularly.
3.1.2 Presenting API links to the teacher

The API query is searched in the different sites listed above as well as in 3rd party search engines to obtain web links. Additional tags (such as [java]) and other filters are added to queries for better results. When no results are obtained, the system looks for results from other sites using 3rd party search engines such as Google. The first five results returned from each website are presented to the teacher who categorizes it into FAQs, tutorials, articles and other web related links as shown in the figure 8. Web links that the teacher feels to be less useful can be removed from the selection by choosing the “Discard” option. The teachers can also rate the links on a scale of 1 to 5. These API related web links are displayed to the user in descending order of rating. On final submission, these choices are packaged along with the current APIs quiz/example selections and saved to the database.

<table>
<thead>
<tr>
<th>Please select the type and order for FAQs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Connecting to remote MySQL database - Stack Overflow</td>
</tr>
<tr>
<td>Hey guys I'm trying to connect to my remote mysql database and my... You need to add the JDBC Connector J driver for MySQL database at..... View FAQ</td>
</tr>
<tr>
<td>Select Item Type: FAQ, Tutorial, Article, Related Links, Discard</td>
</tr>
<tr>
<td>oracle - simple java program connecting to database server using ...</td>
</tr>
<tr>
<td>I've got a very simple java program (java, see below) on my ... Answer is as follows (from Context forums): Try editing /etc/sysconfig/database to..... View FAQ</td>
</tr>
<tr>
<td>Select Item Type: FAQ, Tutorial, Article, Related Links, Discard</td>
</tr>
<tr>
<td>ojdbc - java connecting to access database problem - Stack Overflow</td>
</tr>
<tr>
<td>Top answer: In your connection string there is unescaped : Try with this string: DriverManager.getConnection(&quot;jdbc:odbc:Driver=Microsoft Access Driver (*.mdb&quot;, ..... View FAQ</td>
</tr>
<tr>
<td>Select Item Type: FAQ, Tutorial, Article, Related Links, Discard</td>
</tr>
</tbody>
</table>

Figure 8. Shows the API related links returned to the teacher.

The API related links section is presented to the user once he/she finishes the Quiz/example tutorial for the current topic. The users can also rate different API related links based on whether it was helpful or not. When more people start using the system and with
many ratings, it would help learners to identify the most useful links for each API topic. The rating feature is shown in figure 9 below,

![Related Links and Frequently Asked Questions for current tutor](image)

**Figure 9. Users can also rate each API related link**

### 3.2 Integrating repository code

This feature allows teachers to upload Java code from open source code repositories to jTutors, and students can search for different API usages from these files. We have written a program that downloads Java code from various projects in Sourceforge.net and uploads it to the repository. However, this feature can also be used to upload code from projects of teacher’s choice. With more teachers contributing to jTutors’ repository, the code content grows and we have a lot more API usages to search from. The work flow of this feature is shown in figure 10.
3.2.1 Indexing Open Source Java code to retrieve API code samples

Initially the teacher can upload multiple files/projects to jTutors using the “Code Upload” tab (as shown in figure 11). The files uploaded are filtered based on extensions and only Java files are saved to a temporary folder on the server. If multiple files are uploaded with same name, they are renamed to have a unique file name. Once all the filtered files are saved in the temporary location they are then processed as explained in the next section.
3.2.2 Analyzing Code to Be Indexed

We process the code to be indexed with the help of analyzers, which help tokenize Java code and extract relevant API methods and code segments and import declaration. They can also be used to perform desired filtering and discarding common words before being stored in indexes. However, all that is important to know is that the files are indexed in a format that allows easy querying and faster retrieval of API usage. The Implementation details of indexes are explained in section 3.4.2.

3.2.3 Searching Indexes

After processing the uploaded files and creating indexes for it, they are moved to another location for search retrieval. The indexes can be searched using a custom query which is generated automatically every time a user queries API methods in the search bar shown in figure 12.

For example, if the user queries “getConnection”, then the system will return all documents that contain the “getConnection” in either the “code” or “method” index field.

Figure 12. Student can search for API method usages here.
3.2.4 Scoring Terms

The score of a term is directly proportional to its rarity. jTutors computes scores for indexed items based on a term's document frequency, which is a count of the number of documents that contain the term. The scoring formula can be customized by extending the class `org.apache.lucene.search.Similarity` [44]. The abstract methods in the class are implemented for the scoring of query ‘q’ in document ‘d’

\[
\text{Score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} (\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot \text{t.getBoost()} \cdot \text{norm}(t,d))
\]

where,

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tf (t in d)</td>
<td>Measure of the number of times a term (t) appears in a document (d)</td>
</tr>
<tr>
<td>Idf (t)</td>
<td>Measure of how often the term (t) occurs in other files in the index.</td>
</tr>
<tr>
<td>coord(q,d)</td>
<td>Number of terms in the query (q) that was matched in this document (d).</td>
</tr>
<tr>
<td>queryNorm(q)</td>
<td>Normalization factor used to compare across queries.</td>
</tr>
<tr>
<td>t.getBoost()</td>
<td>Field boost intended to make one field more important than another.</td>
</tr>
<tr>
<td>norm(t,d)</td>
<td>Pre-processed values used for storage efficiency</td>
</tr>
</tbody>
</table>

The important assumptions to be considered in our scoring approach is that,

- Source code with many lines is not as good a result as short ones.
- Documents that contain all the search query terms are a good match.
- Matches for rare words are better than common words.
- Files which contain the searched API terms many times are good search results.
3.2.5 Displaying Code Samples

We have also implemented a simple ranking mechanism for ordering search results, which is in increasing values of scores calculated by the ratio,

\[
\text{(No. of lines in the code)} \div \text{(No. of occurrences of the searched API method in the code)}
\]

This ensures that the source files with fewer lines per API method are on the top of the search results. The search results are also displayed with a small preview showing the first occurrence of the searched method in the file (as shown in figure 13).

![Image of code samples]

Figure 13. Search results page for Industry Code search feature.
Clicking on the file name in the search results opens up the file in a new tab and the code is also prettified with proper indentation, making it more readable to the user. A sample file returned is shown in figure 14 below,

![Sample code returned for Industry Code Search](image)

Figure 14. Sample code returned for Industry Code Search

### 3.3 Cognitive dimensions in design decisions

While designing the interface for the FAQs and Industry Code Search we applied knowledge of cognitive design principles [37].
• **Limit Premature Commitment**

While the teacher categorizes the API web links into FAQs, tutorials and other related articles, the selections are not prematurely saved to the database. Instead the system waits until the user submits the selections before saving it to the database.

• **Visibility**

While displaying the search results of API related code samples in Industry Code Search, the system ranks the search results and shows them one below the other with a preview (as shown in figure 15). The full source code can also be viewed by clicking on the file names and thus making it easily accessible to the users.

![Figure 15. Result preview for the ‘Industry Code search’ feature](image-url)
• **Juxtaposability**

In the FAQs and related links page, all the relevant links are classified based on category and arranged side by side, enabling users to view all the links and their previews in a single screen (shown in figure 16).

![Related Links and Frequently Asked Questions for current tutor](image)

Figure 16. The FAQs, tutorials, Articles and Related links are juxtaposed next to each other.

• **Closeness of Mapping**

The API method entered by the user in “Industry Code Search” is transformed into a custom query syntax that is used to query the index files for results. Queries can be as simple as a single word to as complicated as a nested Boolean expression with conditions added it [38]. A natural programing language style of notation makes things easier to understand and is more closely mapped to the actual result being queried. The code segment which implements the code search query is shown in figure 17.
Eg. A query title: "Head First java" AND text: "JDBC" returns a list of files that have a title index as “Head First Java” and a text index containing “JDBC”. Lucene’s query syntax is intuitive and closely mapped to actual English sentences.

3.4 Implementation Details

jTutors has been implemented in such a way that it can be accessed by anyone with a computer and internet connection. It runs on any system with Java installed and connected to the internet with a browser that supports JavaScript. Additionally, the students may create an account in jTutors to access different tutorials, rate and “favorite” them as well.

The system has been implemented entirely using Java as a dynamic web project. We used Eclipse Java EE IDE to write the code and the system was deployed as a web application using
Apache Tomcat 7.0 ([http://tomcat.apache.org/](http://tomcat.apache.org/)) as the server. SQLite ([http://www.sqlite.org/](http://www.sqlite.org/)) was used as the relational database management system. Model-View-Controller design pattern was followed for programming and thereby components were made to be loosely coupled from each other. The *model* consists of classes that contain the core business logic of the system. The *view* consists of Java Server Pages (JSP) which generates outputs for the user. The *controller* is a set of servlets that communicates with the model and view and ensures that the user inputs are translated into proper operation between them. The MVC architecture ensures that future changes in system dependencies would not change the system logic. JUnit ([http://www.junit.org/](http://www.junit.org/)) was used as the unit testing framework and it allows us to create a base class that can be extended to create a set of tests for the classes you are creating and to evaluate its results. The following subsections explain the implementation details of the new features added to jTutors

### 3.4.1 FAQs and API related links feature

Soon after the Quiz/example selections have been made by the teacher and the Tutorial is described and saved, the system now generates the API related FAQ links. This is done by scraping through links from highly rated Java websites, forums and Google search results. Previous research shows that programming blogs, developer forums and Q&A websites are new ways by which programming languages are documented [39]. These tools help developers to share knowledge and experiences without relying only on official documentation. Presenting relevant links from these sources can be really helpful to familiarize themselves with different APIs and its intricacies. FAQs posted in different sites are an example of crowd documentation and may help to find specific use cases not covered by any traditional form of documentation.
[39]. But the problem is that these links are scattered across different websites and it is not easy to find FAQs without knowing the API question to query. jTutors automates this process by presenting the teacher with important web links related to the current API being queried. These results are then filtered and categorized by the teacher before presenting it to students in an intelligent tutor.

FAQs and related links for different API topics are obtained by querying these sites and scraping relevant links. The HTML tag structure of such websites are analyzed and pages with API related links are scraped using libraries such as jsoup (http://jsoup.org/) and HtmlUnit (http://htmlunit.sourceforge.net/) are used for screen scraping websites. ‘Google custom search API’ (https://developers.google.com/custom-search/) is used in cases where no useful results are returned by scraping form individual sites.

Jsoup is a library used for parsing HTML and it provides a convenient API for extracting and manipulating data in the pages [41]. This is used for obtaining FAQ results from JavaBlogs.com. HtmlUnit is a GUI-less web browser that allows high-level manipulation of websites from other Java code [42]. This library can be customized programmatically to obtained API related links from websites such as Stackoverflow and DZone. Google Custom search API was used while searching for articles in Java2s.com and jGuru.com and retrieving useful online articles. The same approach can be extended to scrape API related web links from other websites as well. The fact that we use different libraries for scraping websites, by itself highlights the need to learn APIs to develop different functionality.
The Top 5 search results from each of these sources are then presented to the teacher as shown in figure 18. The teacher can now categorize the different API related links into FAQs, Tutorials, Articles and Related Articles. Some of the links that might not be very helpful to learn the API or way too complex to comprehend can be discarded by the teacher. The chosen links are then saved to the current tutor topic and presented at the end of the intelligent tutorial, immediately after the quizzes/examples section. The FAQ links and tutorials combined together help learners to know more about API usages and understand them better.

Clicking on ‘Finish’ packages the intelligent tutorial for the current API related topic and posts it to the website. The intelligent tutorial now comprises of a topic description, quizzes, examples, a set of FAQ and API related web links.
3.4.2 Industry Code Search feature

This feature allows users to search API method usages from open source code uploaded to jTutors. Several websites and applications provide features to search all of the information present on the site. These search engines are very effective in retrieving information from web pages and textual content, but are very slow while searching source code. They consider source code files as plain text and hence and are no different from a ‘grep’ tool used to search through text in files. In jTutors we use Apache Lucene, a Java based open source library to search source files by parsing and indexing code elements. The detailed code explanation of this feature is beyond the scope of this report, but this section provides a high-level view of its implementation.

Lucene is not a stand-alone application that can be just installed and used. Instead, it offers us a powerful core API that can be used to add functionality such as text indexing and searching [43]. It also calculates scores for each source file that matches a search query and returns the most relevant results based on these scores. We restrict the search feature to only return Java code. However, it is very easy to extend this feature to support other programming languages as well.

Java code uploaded to the system is tokenized and processed to extract relevant API methods, code segments and import declaration.
Two common analyzers provided by Lucene are [43]:

*SimpleAnalyzer:* They can be used to tokenize the entire code to sets of strings.

*StandardAnalyzer:* They can be used to tokenize the strings to sets of words by identifying API methods, declarations and other code segments.

The analyzers are implemented to process code in two stages as explained below,

*Tokenizer* splits code into text. For example, StandardAnalyzer use spaces and punctuation to tokenize code into separate chunks.

*Stop words* are used to detect non Java chunks in code, which comprises of English words like "I", "the", "is", "was" etc. Stop-words filter removes them from the chunks generated by the Tokenizer. This helps to lower the noise in search results and return better code samples.

The index files thus generated can be viewed by an open source tool called “Luke” ([http://code.google.com/p/luke/](http://code.google.com/p/luke/)) which is available for download as a jar file at ([http://www.getopt.org/luke/](http://www.getopt.org/luke/)). The teacher can use this tool to browse indexed files, search, delete and modify index fields. A screenshot of Luke’s interface in figure 19 shows the different fields and values of a sample Java file as saved in the index file.
The API method entered by the user is searched in these indexes and the resulting files are returned to the user. Lucene’s custom query syntax is used for querying these indexes and it also provides a lexical analyzer which interprets a string into a Lucene Query using JavaCC parser.

Figure 19. Luke is eased to for viewing and modifying index files generated.
4 Evaluation

This section focuses on the Qualitative evaluation of a User Study which we conducted in a laboratory setting to evaluate the new features added to jTutors. The tutoring methodology implemented in jTutors is supported by three main features.

- Quiz and example tutorials (already evaluated)
- FAQs and related links (new feature)
- Industry Code Search (new feature)

A problem that we faced when conducting our preliminary quantitative study with jTutors was that it was challenging to know the user’s learning experience and other human aspects of using our system [47]. Evaluating our system with a purely quantitative approach might not be effective in this case because it is difficult to quantify what users feel while using our system. Thus we decided to evaluate our new features using a qualitative approach. This type of research is usually used for investigation of situations where people are involved and different processes can take place. This would help us learn about the user’s experience and situations that cannot be retrieved using quantitative data analysis methods [48]. Qualitative study results can illuminate new insights into system effectiveness [10].

One of the main goals of qualitative research is that instead of aiming at accepting or rejecting a pre-defined hypothesis, it is aimed at constructing a theoretical framework that emerges from the data gathered during the user study [48]. This framework is called “Grounded Theory” and it can help in generating a theory from the data obtained during the research user study [9].
The process of collecting data using qualitative research typically includes interviews, surveys and observations. As a result of such a process, the study usually focuses on a relatively small number of participants [48]. In order to have a level of generalization in qualitative research results, participants are carefully selected and the descriptions of the results are very detailed and specific [48].

4.1 Study Methodology

Our goal in this user study was to evaluate the effectiveness of new features in our interactive training materials for helping students understand APIs. Intelligent tutors were created for 5 topics that were reported as being typical queries posed by users in the MICA search engine (retrieves code snippets online showing how to invoke Java APIs) [15].

We recruited 17 students from Oregon State University with novice level programming experience for the study. The participants mainly consisted of students taking undergraduate level programming courses. The process by which we conducted the user study is explained:

Learn a Java Topic (15 minutes)

The user was shown a demo of jTutors and its features and then allowed try the system for 10 minutes. He/she was then asked to learn a Java topic using our system. A series of examples and quizzes related to API topic were presented to the user while learning. Quizzes were very similar to the examples. The only difference was the presence of blanks. If the user typed an answer in the blank and pressed enter, the box turned green if correct and red if incorrect. The “Get Hint” button was used to get hints regarding the method name to be entered in the blank. The user was given 3 hints for help. The first two hints helped with the
answer and the 3rd one was the answer itself. These tutorials are only for learning and not a test of user’s knowledge in the domain. Field notes are recorded down during the actual observation; after the observation is over, notes are filled in with as many additional details as one can remember. The time duration for each user to work on different sections of the system is also noted down.

Task 1 – FAQs and Related Links feature (20 minutes)

The user was given a sheet with questions related to the API that he learnt. He/she was allowed to answer these questions with the help of jTutors. The user answered these questions based on what they knew about the API, or used the FAQs section to check related links. In the “FAQs and Related topics” section the user was presented with links to FAQs, tutorials, articles and other links related to the current topic. This task was a way to both use the “FAQ feature” as well as learn from it. The observation data as well as interview responses helped us know about the usefulness of this feature.

Task 2 – Industry Code Search feature (20 minutes)

The user was given a Java source code with API related methods blanked out. A set of API method names was provided as options for the blanks to be filled up. In cases where user was not sure about which method to use and where, he/she used the “Industry Code Search” feature to check for the usage of different methods in other programs. Searching for API methods in the “Industry code” search bar returned occurrences of the method in the repository files. Clicking on individual results showed the file’s entire code. The user then filled
out the answers using assistance from the system. This task was a way by which they got to use the “Industry Code Search” feature as well as learn form it. The observation data and interview helped us know about the usefulness of this feature.

**Interview**

In interviews, data are gathered interactively where the interviewer asked questions related to the learner’s experience, while a scribe took notes of the conversation. We employed a semi-structured interview pattern for this purpose, which was a mixture of open-ended and specific questions designed to know both foreseen information as well as new insights about the system [47]. Qualitative interviews can also generate quantitative data depending on the ways in which they are combined with survey techniques. There were a set of questions in the “Interview Guide” which was asked to all participants. The interviewer also asked other questions to get more information from the user’s learning experience.

The questions in the *interview guide* are given below,

I. Tell me about your experience with jTutors.

   *Question’s Focus – Usefulness/Experience*

II. What did you like most about jTutors?

   *Question’s Focus – Opinions/Values*

III. What do you think about the usability of the system?

   *Question’s Focus – Ease of Use*

IV. If there is something that you would like to improve in this system, what is it?

   *Question’s Focus – Feelings/Ideas*
V. What did you feel you learned from being a part of this study?

*Question’s Focus – Knowledge*

VI. Is there anything else you would like to tell me about jTutors that was not covered in the previous questions?

*Question’s Focus – Other Comments*

Often we needed additional information from the user during observation or while interviewing. In such cases we employed a technique called probing where we asked follow up questions to fully understand the user’s response or when we need in depth information on some of their actions.

*Survey*

The user was given a survey form with a set of questions which asked to evaluate the system’s features in a rating scale of (1-5).

- How would you rate your experience learning APIs using jTutors? (1 to 5)
- How would you rate jTutors feature to check FAQs and other API links? (1 to 5)
- How would you rate jTutors feature to check code samples from actual Industry code? (1 to 5)
- How would you rate the ease with which you use and understand the tutoring style of jTutors? (1 to 5)
- If our instructional materials were available for your courses at Oregon State University, would you use them? (1 to 5)
Finally the user was paid $20 for participating in the study and sharing feedback about the system. A signed receipt was collected for the money that they received.

**Data Verification**

The quality of qualitative research lies mainly on how the data is gathered and analyzed [14]. Validation was done by checking the field notes with a sample of 4 participants for the accuracy of the data taken note of during observation and interviews. The field notes were found to be accurate and there weren’t any changes suggested by the participant.

Two data collectors were involved in the entire process of participant observation and interviewing. Other data sources such as logged data of user clicks and screenshots of user actions were used to verify data. We captured user actions by taking screenshots at regular intervals and during keystrokes using “Wink” (http://www.debugmode.com/wink/), a free software to take screenshots. Data obtained from these screenshots helped us in verifying some of the observations we recorded in our field notes. The screenshots are time logged and hence it also helped us know the amount of time spent in different sections. Verifying the results increases the trustworthiness of the study.
4.2 Analysis Methodology and Results

Qualitative findings are usually generated by an inductive process where detailed information obtained from the study is classified into general themes [7]. A thematic analysis of the data is carried out. It typically involves -

a. Reading the scribes/notes of the study and identifying patterns/themes.

b. Re-organizing the data based on patterns and coding them.

4.2.1 Deriving the Code

Coding is a strategy used to combine data into categories and mark similar passages of text with a code label so that they can be used for comparison and analysis. Our study was conducted by two researchers and both of them coded the responses for the qualitative interview individually and then compared similar ones, discussed and then coded them again.

A test of inter coder reliability can reduce errors that may result from inconsistencies between coders [45]. The level of reliability is determined by the “Intercoder Agreement Coefficient” [13]. This is given by the formula:

\[
\text{Intercoder Agreement Coefficient} = \frac{\text{Agreements}}{\text{Agreements} + \text{Disagreements}}
\]

Krippendorff [46] set a rule of thumb that if the coefficient value of above 80% then it is deemed good and if between 67-79% then it’s acceptable. Riffe, Lacy, and Fico [12] stated in their paper that coefficient values below 70% are difficult to interpret. One downside of the
simple percentage agreement calculation is that there it does not account for the possibility that coders may agree by chance [13].

In our study we had an agreement coefficient of

\[
\frac{26}{26 + 4} = 0.866 \text{ (ie. 86.6\%)}
\]

Where agreements = 26 and disagreements = 4, while coding.

This is a good measure indicating that the coding is reliable as per Krippendorff’s rule [46].

Given below are the results of coding the qualitative interview responses and quotes by some users about jTutors -

**Qn 1. Tell me about your experience with jTutors?**

<table>
<thead>
<tr>
<th>Codes</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning By Example helped</td>
<td>11</td>
</tr>
<tr>
<td>Felt Good</td>
<td>7</td>
</tr>
<tr>
<td>Easy to Learn</td>
<td>4</td>
</tr>
<tr>
<td>Helps to Practice</td>
<td>1</td>
</tr>
<tr>
<td>Felt repetitive</td>
<td>1</td>
</tr>
</tbody>
</table>

One of the participants’ comments on the system was that "They are very easy to learn from and much better than using API documentation". Another user had a contrasting opinion and said “Quizzes are fairly repetitive. Overall it was okay”.
Qn 2. **What did you like most about jTutors?**

The responses for this question have been coded and classified based on features that they liked the most and their opinions about them.

<table>
<thead>
<tr>
<th>Codes</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tutorial Section</strong></td>
<td>7</td>
</tr>
<tr>
<td>Instant Feedback</td>
<td>1</td>
</tr>
<tr>
<td>Easy to Learn</td>
<td>1</td>
</tr>
<tr>
<td><strong>FAQ/Related Articles Section</strong></td>
<td>6</td>
</tr>
<tr>
<td>Exposed common issues</td>
<td>5</td>
</tr>
<tr>
<td>Api info - All in one place</td>
<td>2</td>
</tr>
<tr>
<td><strong>Industry Code Section</strong></td>
<td>10</td>
</tr>
<tr>
<td>Helps search sample code</td>
<td>10</td>
</tr>
<tr>
<td>Helps Learn Syntax</td>
<td>1</td>
</tr>
</tbody>
</table>

It can be seen from the pie chart below that majority of the participants liked “Industry Code Search” the most when compared to all other features in jTutors.

![Pie chart](image)

Figure 20. Pie chart shows the distribution of features that users felt was very useful
Participants mentioned that the length of the code snippets in quizzes/examples was ideal. A user quoted about FAQ feature that it “Exposed what issues people commonly face”. Another user liked the Industry Code Search feature and said “When I use Google it takes a while to actually find good examples, which I can use to understand methods. Industry code search provides a good and easy way for that”.

There were many users who felt the Industry Code search to be highly helpful and given below are some of their quotes –

- "It can be very useful to learn something unfamiliar. This would be helpful with syntax and usage"
- "Instead of first looking through API documentation, Industry Code Search helps by showing method usages"
- “Provides examples of different usage scenarios. It helps my style of learning which is learning by example"

Qn 3. **What do you think about the usability of the system?**

Most of the participants felt that the system was simple and user friendly. A few of them also mentioned that the system was less distracting and that was ideal for them to gain knowledge using the system.

<table>
<thead>
<tr>
<th>Codes</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Friendly</td>
<td>13</td>
</tr>
<tr>
<td>Simple/Intuitive</td>
<td>10</td>
</tr>
<tr>
<td>Not Distracting</td>
<td>7</td>
</tr>
<tr>
<td>Inconsistent</td>
<td>3</td>
</tr>
</tbody>
</table>
A participant said that the system is "Easy to use and simple. This is faster compared to
hour long lectures I go to. It's also less distracting"

Qn 4. If there is something that you would like to improve in this system, what is it?

The users had a bunch of improvements to suggest regarding the system. Some of them
mentioned that they would want to see more API related information and descriptions of
quizzes/examples shown in the tutorial. One of the participants suggested a tweak in the way
descriptions are shown by saying that "It would be good to have incremental descriptions about
the API methods in the tutor rather than a detailed description in the beginning". A couple of
participants felt that having a dictionary of API methods displayed while using an API tutor
would help them to learn from it as well.

<table>
<thead>
<tr>
<th>Codes</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>More API Descriptions/Comments</td>
<td>8</td>
</tr>
<tr>
<td>Dictionary of API methods</td>
<td>2</td>
</tr>
<tr>
<td>Nothing</td>
<td>2</td>
</tr>
<tr>
<td>API recommendations</td>
<td>1</td>
</tr>
<tr>
<td>More complex tutorials</td>
<td>1</td>
</tr>
<tr>
<td>Learning Progress Indicator</td>
<td>1</td>
</tr>
<tr>
<td>More feedback</td>
<td>1</td>
</tr>
</tbody>
</table>

A participant suggested that he would like jTutors to have a feature for “Being able to
show recommendations for related/similar APIs". Another user said, "Looking at hints in
tutorials makes me feel that I could get through the tutor without learning anything". This
clearly is a sign that this mode of learning wasn’t useful for all learners. Interestingly it must be
noted that the participants who felt that this mode of teaching wasn’t helpful were programmers who had previous knowledge about API they learnt using jTutors. Hence, it can be said that the system works more effectively for novice programmers than with those who have prior experience using Java APIs.

**Qn 5. What did you feel you learned from being a part of this study?**

Most of our participants strongly felt that they were able to learn the API topic by using jTutors. Some of them made a comparison saying that they would prefer to use jTutors for learning instead of textbooks.

<table>
<thead>
<tr>
<th>Codes</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learnt the API</td>
<td>13</td>
</tr>
<tr>
<td>Preferred jTutors to textbooks</td>
<td>5</td>
</tr>
</tbody>
</table>

A participant quoted, "It would be good to have this to learn from. This seems much more easier to learn Java than lectures or books". There were also comments about jTutors features and an user said that “Industry Code feature would be beneficial to people doing own project to get to know more information in a selective manner”

**Qn 6. Is there anything else you would like to tell me about jTutors that was not covered in the previous questions?**

Participants were free to comment on anything related to the system, its features or their experience as a whole.
<table>
<thead>
<tr>
<th>Codes</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industry code search more helpful than Google</td>
<td>5</td>
</tr>
<tr>
<td>Lots of info to sift through online</td>
<td>3</td>
</tr>
<tr>
<td>Length of code in tutorial was ideal</td>
<td>2</td>
</tr>
</tbody>
</table>

Around a third of them spoke about how they felt industry code would be useful for them to find code samples while learning APIs or coding for projects. Given below are some of the quotes related to it -

- "Google is kind of hit or miss when it comes to conceptual or code usage questions. I think this system fares well in that respect."

- "While using the internet, I feel there is a lot of information to sift through, but this system provides relevant information all in one place."

- "Searching for method names on Google for finding code samples works but it takes 4 or 5 links in the results to get to it. The Industry Code Search feature works much better in this respect, I can usually find information from just the preview in Code Search results."

- "Google for code search and code snippets is a long process and very often the results are not what I am looking for. Industry code search is definitely better than simple Google search for code search."

- "Google search results usually present API docs mainly. They are good to know what a method is but to search for actual usage a lot of results have to be gone through. Industry code search is much easier to use in this way."
4.2.2 Other Lab Observations and Survey Results

Field notes taken during the user study provided valuable insights into certain usability aspects of the system and its behavior which would have been otherwise difficult to find out with just study tasks. The results of the field notes was triangulated and verified between the two observers after each study session. Some of the interesting observations are listed below:

- Users found it easy to answer fill-up API methods in blanks (Task 2) by looking into the code examples of different methods in real world programs. In most cases, just looking at the search result preview helped them understand the different usages of methods.

- Some of the participants felt the urge to go back and forth during quizzes to check for the previous example or description to answer the current quiz. Since the system is computer adaptive and the quizzes/examples are show based on the user’s current learning progress, the interface has to be tweaked in future, to make sure users can’t go back to the previous page and seek the help of previous examples/quizzes to answer the current one.

- While using FAQs and related articles, users tended to check for the source website of different articles before reading through them. It can be understood clearly that people preferred links from certain websites when compared to others.

- Some of the participants went through all the links in the FAQ section although only a few of them helped them with their task. The users really showed the urge to know more about the API by browsing through different links.
The participants also filled in a survey of 5 questions with the scale given below for rating:

<table>
<thead>
<tr>
<th>Not Useful</th>
<th>Good/Neutral</th>
<th>Very Useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

The survey questions as well as the mean scores are listed below:

- **How would you rate your experience learning APIs using jTutors?**
  - *Mean Rating: 4.29 / 5.0*

- **How would you rate jTutors feature to check FAQs and other API links?**
  - *Mean Rating: 4.24 / 5.0*

- **How would you rate jTutors feature to check code samples from actual Industry code?**
  - *Mean Rating: 4.53 / 5.0*

- **How would you rate the ease with which you use and understand the tutoring style of jTutors?**
  - *Mean Rating: 4.29 / 5.0*

- **If our instructional materials were available for your courses at Oregon State University, would you use them?**
  - *Mean Rating: 4.53 / 5.0*

The results of the survey indicate that this approach of learning Java APIs was very useful for the users and it can also be noted that people rated the “Industry Code Search” feature more than the “FAQ and related links” feature.
5 Conclusion and Future Work

Our research contribution through jTutors is to provide new features to help users learn about how APIs are used in practice apart from different methods and their usages. The new features present API related web links, FAQs and code samples from open source code repositories, making jTutors a one-stop shop for learning different aspects of any Java API.

The FAQs and related links section helps learners to familiarize themselves with most commonly made mistakes with using APIs, to know other APIs with similar functionalities and to read API related articles. This will be of great help to users as there is currently no tutoring system which shows API related FAQ links. However, Google is more preferred in cases where the user knows clearly what question he is looking for online.

‘Industry Code Search’ can be used to check API code samples in actual open source projects, thereby helping learners to know how different APIs are combined to create functionality. Software engineering companies could also adopt this approach to teach new hires about how to use proprietary APIs and familiarize themselves with its usage in company projects. This feature has been compared to be better than Google search for codes by many study participants, but there is still more scope for improvement. The feature could be extended to support API specific method search by means of specifying the jar files or package imported etc.

Our approach encapsulates API knowledge in intelligent tutors which we expect to become durable instructional assets for use in training students via the web. In particular these tutors
could be used as the basis for teaching Java API concepts in introductory computer courses as well as long distance programs for professional software engineers.

Qualitative study results clearly indicate that users found the new features very useful to learn APIs, especially novice programmers. The system provides a new way wherein all the learning materials for Java APIs are leveraged from web links, open source code repositories and semi automatically packaged into intelligent tutors.

The role of the teacher in creating jTutors is very critical. Although the system suggests the code samples/API related links, it is up to the teacher to make selections in order to make it a good learning material for the students. Creating tutors involves selecting quizzes/examples, providing hints, descriptions and categorizing API related links which requires quite a bit of time as well. The system right now takes about 20 minutes to create an entire tutorial and this can be speeded up in future by adding indexes to the API terms stores in the database. This helps in finding relevant examples more quickly and efficiently.

In the future, jTutors website could be further enhanced to become a platform for sharing tutorials among the Java teaching community. A fellow researcher is already working on this aspect of jTutors. Such a website could help users to share their tutorials with other students and users. Currently, jTutors does not support options to edit tutorials once they are created. In future, the system could allow the users to customize the tutorial and share it again. Allowing users to comment on the tutorial would help others to read reviews before trying out a tutorial. The sheer number of online communities dedicated to discussions on Java shows the popularity of the language and willingness amongst novices to learn the language. The effectiveness of
such a platform can be evaluated by an empirical analysis of the user’s interaction with the website.

Currently we are providing tutorials as a way to solve quizzes or read examples with some hints provided by the teacher. But it would be an interesting project to make these quizzes show more explanation about the code snippet presented, based on the user’s request. This can be achieved by extracting the text or images displayed on the webpage from which the code snippet was scraped. The jTutors idea can also be extended to create a curriculum planning system for teaching Java APIs. Instead of generating only examples and quizzes, the system could be tailored to generate chapters that teach a Java API. Since the system can test the knowledge of students with the help of quizzes, this information could be used to create a model of the student. This information could be used in judging the student’s current proficiency and help in recommending new topics to learn.

jTutors uses snippets from the web that pass the error-tolerant parsing algorithm implemented by the Eclipse AST API. The teachers could use their judgment to create more elaborate quizzes from these snippets. This would mean, instead of blanking out one word at a time, the teacher could blank out a text area and annotate it with comments explaining the purpose of the lines of code that should go in the blank area. The students would then be required to write a solution to the given task rather than only filling out single words. This would help the teacher to test the student’s understanding of different APIs. The Eclipse AST API can then be used to find the student’s syntax errors and generate constructive feedback in the form of error messages and suggestions. Since jTutors is a highly interactive system, its success would depend on feedback from user studies on teachers and students.
The work done in creating jTutors could be extended to other domains as well. Similar to Java, other programming languages like C#, PHP, etc. also rely heavily on APIs for providing all the exciting features to the programmers. It would be interesting to adapt jTutors to different languages. More laboratory testing with teachers and students will help to improve the effectiveness and ease of use of the system. In the cases where the results returned are not useful, further analysis should be done to identify the reasons.
6 Bibliography

[1] What is Java technology and why do I need it?
   

   of the Fourth International Workshop on Computing Education Research, 3-14.

   [Statistics cited are based on Figure 1]

   development job. Proceedings of the 39th SIGCSE Technical Symposium on Computer
   Science Education.

   opportunistic programming: Interleaving web foraging, learning, and writing code.
   Proceedings of the 27th International Conference on Human Factors in Computing
   Systems.

   Software Engineering, 1-30.


   Political Constraints.


   qualitative research.


[13] Intercoder Reliability


http://jtf.acm.org/rationale/rationale.pdf


http://www.digitalwell.washington.edu/rcuwtvdownload/uw_cse05_inteltu_ipodv.m4v
(retrieved on May 24, 2012)


[38] Understanding Lucene’s query syntax


[41] jsoup: Java HTML Parser

http://jsoup.org/ (retrieved on May 24, 2012)

[42] HtmlUnit - From Wikipedia, the free encyclopedia


[43] Apache Lucene

http://onjava.com/onjava/2006/01/18/using-lucene-to-search-java-source.html
(retrieved on May 24, 2012)
[44] Customizing Lucene's scoring formula

(retrieved on May 24, 2012)


[48] Hazzan, O. Qualitative Research in Software Engineering