How Programmers Use Internet Resources to Aid Programming

Jeffrey Stylos
Computer Science Department and Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213, USA
{ jsstylos, bam }@cs.cmu.edu

Brad A. Myers

ABSTRACT
When programmers create new software or add functionality to existing software, a common and often difficult task is that of figuring out how to use libraries, toolkits and SDKs to achieve the desired functionality. Internet resources, in particular Google, have emerged as new and effective tools in this task, providing quick access to a large collection of tutorials and example code. However, there has been little study of how programmers use these Internet resources and how these resources could be designed to better aid programmers. In observations of programmers we find that their web queries can be classified into two categories: high level searches to find tutorials and available methods, and lower level searches to find code examples that use particular methods. In addition, we observe a tendency to iterate between these two types of searches as programmers refine their knowledge of the libraries and methods. Given our observations, we suggest new types of programming support tools, such as IDE features to help programmers understand and run code snippets found online.

Author Keywords
Programming help systems, Internet-based help systems, Google, information seeking.

ACM Classification Keywords

INTRODUCTION
A fundamental programming activity is that of creating new software functionality, either as a new application or an extension to an existing application. For non-trivial functionality, this usually requires using a collection of libraries, toolkits, APIs and SDKs to achieve the desired functionality. Learning how to use these libraries is an essential part of transitioning from novice to intermediate programming, and a significant part of the challenge in starting a new project and contributing features to existing projects. Expert programmers also frequently have to learn new APIs, for new programming projects and when new versions of SDKs are released.

There are many tools to help aid programmers in learning how to use libraries, including written documentation and interactive help systems. However, a new and popular strategy for programmers is to use resources on the Internet, such as forums and search engines, to find tutorials, examples and answers to programming questions. These Internet resources often lack many of the advantages of more traditional help systems, such as well written, verified information and advice, but offer advantages such as enormous databases of content and intelligent search algorithms to make the most useful and relevant information easiest to find.

To better understand how programmers use these Internet resources, and when they are effective and when they are not, we studied programming projects in various stages of creating new functionality and observed how they used Internet resources to support their programming. In these case studies we find common patterns of usage, and different situations where the effectiveness of these tools breaks down. These observations suggest the design of new programming support tools that take advantage of, extend, and fix some of the problems with how Internet resources are used for programming support.

RELATED WORK
There has been much research on information seeking and software documentation.
The information seeking literature includes some studies of how programmers [1] and others [2][3] use internet resources. However, even when focusing on programmers, these mostly look at more formal documentation, with an emphasis on implications for documentation writers. Our observations include the use of many informal resources, such as forum and Usenet posts and independent websites with sample applications. Since these sources are distributed and not created by the library maintainers, these bring up different issues, with different implications for creating new tools.

While Usenet, forums and other Internet resources have been around for several decades, we are interested in how modern tools such as Google have made these resources significantly more useful by collecting them, making them accessible and by providing advanced searching techniques.

THE PROGRAMMERS WE OBSERVED
We observed three programming projects in Java, each with one programmer. The programmers were graduate students at CMU who were completing the projects for others reasons, and allowed us to observe.

The first application was a simple Java application by a novice programmer. The program was a currency converter, which used text- and selection-boxes to allow users to convert an arbitrary value in one currency to another. The project required the use of only the standard Java libraries.

The second application was a plug-in for the Eclipse platform. This plug-in was written in Java by an experienced programmer, and was to add a new form of refactoring to the Eclipse IDE, though we were only able to observe the final completion of the plug-in. The plug-in made use of standard Java and Eclipse-framework APIs.

The third application was a modification to an open source project required the use of only the standard Java libraries.

Because of the limited size of the study, one must be cautious before generalizing the results to additional programmers, libraries and languages. However, we have informally observed many of the same behaviors in ourselves and our colleagues and believe that the implications are useful.

OBSERVATIONS
A Model of How Programmers Learn APIs
In observing the programmers, we noticed several different stages of API learning and transitions between these stages. Figure 1 shows these stages and how they relate to each other.

Each of the programmers started with an initial idea of what their application was to do (A). Only after getting an overview of the structure of the APIs they would use (B) did the programmers begin to design how they would implement their application (C). These initial design ideas would sometimes require further high-level API understanding (B). Once they had a high-level design, their next task was finding the specific methods that could accomplish their task (D). Having found the name of a method, they then searched to find exactly how to use it, using documentation and example code (E). They would then integrate this example code into their own program and see if it accomplished what they wanted (F). If it didn’t, they’d find new examples of how to use the same methods (E), look for new methods (D), redesign their architecture (C) or look for different APIs (B).

Programmers used tutorials, articles and example programs to help get an overview of API capabilities in step (B). They used Javadoc documentation, forum answers and example programs to find specific API methods (D) and how to use them (E). Finally, the Eclipse platform provided some help in integrating example code (F) by detecting library dependencies and auto-formattin the pasted code.

Internet Resources Used
Each of our observed programmers used Google as their primary resource for finding programming information on the Internet, with one also using Google Groups, Google’s search engine for Usenet archives. The programmers found a variety of different types of resources using Google, including tutorial pages such as those on http://java.sun.com, documentation such as the Java SDK Javadoc pages, overviews and articles on software architectures, webpages with example programs, and forum posts with questions, answers and code snippets.

High-level API understanding (Step B)
Because the programmers we watched already had a good idea of what program they wanted to create (A), the first step we observed was that of getting an overview of which APIs they needed to use and those APIs’ overall structure (B). Each programmer’s first step in this task was to pose a general query to Google. For example, the programmer writing a refactoring plug-in for Eclipse searched for “refactoring plugin eclipse”. Google was effective at finding tutorials, high-level articles (such as those provided by IBM and Eclipse.com about Eclipse) and sample
projects with source code and documentation that the programmers found useful.

One of the reasons that Google was effective at this task was because it was worked well even with the use of non-expert terminology. For example, the novice programmer creating a simple Java application was able to find information about creating windows and widgets using the search “creating a form in java,” even though the word “form,” which the programmer was familiar with from Visual Basic, is not often used in Java programming.

Once programmers had found tutorials and articles that they thought would be relevant, they would skim them, then leave the web browser windows open as they looked at other search results. Two of the programmers used the tabbed-browsing feature of Mozilla Firefox to group related search results together and more easily switch between their results and the developing environment.

Measured by time, the relevant tutorials and articles were programmers’ most used Internet resources. When they had verified that a tutorial, article or sample project was useful, they kept that browser window open and referred back to it throughout much of their entire programming project. The programmers sometimes used their browser’s bookmarking feature to remember these sites, but when resuming programming after a break or interruption, there was still an overhead in re-finding the relevant sites.

A tool, built into an IDE or web browser, that could automatically keep track of the most used web pages for a given project would reduce this overhead of resuming or switching between programming projects. In addition, creating a long-term association between these web sites and the project would provide a form of documentation that could aid others in understanding the programming project later.

Discovering which methods to use (Step D)

When programmers had begun to understand the high-level elements of the APIs they were using (B), they then formed an idea of how they planned to implement their application (C) and their next action using Internet resources was to determine what specific classes and method calls they could use to accomplish specific tasks (D). Often this step was combined with the previous step (B) as the tutorial or article would include specific method references.

Programmers used different strategies in this step, including browsing the list of classes in the JDK’s Javadoc documentation and searching Google with a description of the desired functionality. Because Google indexes many documentation sources, including the JDK’s Javadocs, searching with Google could often double as a search of the official documentation. In addition, when programmers found a potentially useful method, they would often look up its official documentation to verify that it did what they thought.

A tool that used a large database like all the webpages searched by Google to collect the most common keywords associated with a given method, and the methods associated with given keywords, could build an index that would allow better documentation searching, even when offline. This would take advantage of the ability to use common terminology that Google supports while going directly to the official documentation, which the programmers often referenced anyway as a separate step.

Finding examples of how to use a method (Step E)

Once programmers had found a method they thought might be useful (D), they then looked for specific code examples of how to call the method (E). This was usually done by searching Google, or Google Groups, with the name of the specific method, but was also sometimes combined with the previous step (D), or the previous two steps (B)(D) when the search results already included code samples.

The Javadoc documentation usually did not provide examples of code use. Examples were used to answer such questions as: “How do I instantiate an instance of this method’s class?”, “How do I get variables of the appropriate types to pass as arguments?” and “At what point in my code should I call this method?”.

The examples the programmers found were sometimes in the form of complete programs with a few lines of interesting code, and were sometimes small code snippets without an accompanying full project. The later was usually the case when using Google Groups.

Google can be less effective at this task when the method name is also word or is shared across different libraries and languages. A search engine that could be asked to identify code within webpages and its programming language and library references could avoid these problems.

Programmers also sometimes wanted to search for specific types of uses of a function, such as examples of how to use a function dynamically at runtime as opposed to during initialization. A search engine that could provide a set of search mechanisms based on the semantic nature of the code could be more useful in these cases.

Integrating code examples (Step F)

After finding code examples that used a specific method (E), the programmers would then attempt to integrate that example into their own project (F). Most often this was performed by copy and pasting the code from the web browser to the code editors, though occasionally the code was manually retyped instead. Previous research has also observed a tendency of programmers to copy and paste code and its problems [5].

In the majority of the cases, simply copy and pasting the code into their project did not give them a compilable result. This was because the copied code required import declarations that were not included or not copied, the copied code used variables that had not been declared, or
the copied code contained typos or other errors. The Eclipse programming environment helped with the problem of imports by suggesting which imports to use automatically when they were in the standard Java libraries, though it only suggested these after creating an error flag and if the programmer knew to click on the error flag for suggestions.

Even after fixing compilation problems, programmers often still had problem with the new code. This was because they had not pasted the code in the correct place (and so it was not being called, or not being called at the correct time), the copied code was missing other necessary statements, such as initialization method calls, or when adapting pasted code into their own program, programmers had changed aspects of the code they had not realized were fundamental. For example, when copy and pasting code from a sample Java applet, the beginning programmer had not realized that the “init()” method was a keyword with special behavior.

Because programmers encounter many problems with this task (F) and because it is the transition point between Internet resources and the developing environment, it provides motivation for several new useful types of IDE features.

First, there are features to help programmer paste code from other sources, like webpages. When this code is not directly compilable, the IDE could suggest fixes such as declaring and initializing variables of the appropriate types when they don’t exist. In addition, if the IDE had knowledge of different methods’ dependencies, it could flag uses of those methods that were not in the correct place or that lacked the necessary prior initialization. These dependencies could be specified directly by the library creators or could be automatically generated given a large set of example programs. And an IDE also could flag when newly created code was not being called to help narrow down the debugging problem [3].

The IDE could also contain features that aid programmers in copy and pasting code from one project to another. Copying code from a source file can encounter the same problems as copying from web pages, but because the IDE has more knowledge, it can provide greater support in identifying dependencies and suggesting fixes.

Our observations also motivate features to assist programmers in copying their own code and pasting it elsewhere, such as to provide answers to a web forum. By detecting and including dependencies not directly selected in the copied code, such as imports, variable declarations and the specific location of the code if important, an IDE could automatically include these things in an optional, more complete pasted version. While not as direct a solution as features to help users integrate code, by improving the quality of the code fragments posted on forums and elsewhere, it would eventually reduce the overall problems programmers have integrating code examples.

Reverting back to earlier steps (Steps A-E)
When the examples that programmers found failed to do what they wanted, or were only partially useful, the programmers then reverted back to any one of the earlier steps, looking for new examples of how to use the same method (E), new methods (D), new understanding of the high-level APIs (B) or a redesign of their program design (C) or specification (A).

Given a limitation or problem using a method or API, Google was sometimes useful in helping to decide which step to return to, by providing anecdotes of others who had run into the same problems. This was useful in cases where certain functionality was impossible, or where there were complicated work-arounds, neither of which are usually addressed in official documentation. However, a problem in this case was that Google often returned results with other people asking the same (unanswered) question before returning an answer. A search engine that could differentiate between questions and answers, and – ideally – link the two when related, could provide more useful results.

An additional problem with using Google when returning to previous steps was that additional searches tended to find more results with the same information, rather than new types of solutions. This was partly because the terminology and method names the programmers had learned reinforced the likelihood of finding same pages they had already found and made it difficult to find new terminology and results. A search tool that could separate results into different ways to accomplish the same task could make it easier for programmers to try different solutions when one does not work. These could be displayed as clustered search results, such as provided by the Vivisimo search engine, but would likely need different algorithms for semantic clustering.

CONCLUSION
Our observations of programmers show that Internet resources, especially Google, are commonly used in programming projects involving new APIs. While useful in a variety of different situations, programmers using these resources encountered several different problems. We plan to follow up our research with more in-depth studies of more programmers and to build tools that will address some of the issues we have found, and hope that our observations will motivate related research by others.

ACKNOWLEDGMENTS
We would like to thank Andrew Ko for his ideas and comments on this paper. This work was partially supported under NSF grant IIS-0329090 and by the EUSES Consortium via NSF grant ITR-0325273. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the NSF.
REFERENCES