Developing Best Practices for API Reference Documentation: Creating a Platform to Study How Programmers Learn New APIs

Robert Watson
University of Washington
rbwatson@uw.edu

***Abstract* – Software developers use application-programming interface (API) documentation to learn how to use the features of software libraries. How quickly software developers learn to use a library's features determines how quickly they can apply those features in a software application. Recent studies have shown that API documentation is, unfortunately, not always as helpful to software developers as they need it to be. This paper studies the prototype of a tool and a method that are being developed to help technical writers identify the elements of API reference documentation that help software developers complete programming tasks. The tool and method described in this paper use a remote user-assessment platform, which enables researchers and technical writers to study the effect that document design variations have on a large and diverse audience. Such an approach can help technical writers identify new best practices for writing effective API documentation.**

Index Terms – API documentation, best practices, remote user assessment, technical writing.

INTRODUCTION

Document design affects the readability and comprehension of online and print documents in general, so it makes sense that document design would influence learning in the specific case of the application-programming interface (API) reference documentation that software developers use. Unfortunately, current API reference documentation, the format of which has remained largely unchanged for the past several decades, does not seem to be working for today's software developers as some recent studies report [1, 2, 3]. One study found "that some of the most severe obstacles faced by developers learning new APIs pertained to the documentation and other learning resources" [3].

How can we improve documentation such that it is no longer "the most severe obstacle?" The obvious, but not necessarily the easiest, first step toward answering that question is to define what we should expect from improved documentation. I suggest that, at the very least, it should no longer be an obstacle to learning and, ideally, it should provide a positive return on the reader's investment—that is the software developer should save more time by reading the documentation than they would otherwise spend by not reading it. To accomplish this, it is important to start by having the right content. Robillard and DeLine [3] describe some of the content that software developers feel API documentation needs as do Nykaza, et al. [4]. The right content should then be presented in an efficient format that is easy to read.

It might not seem like there is much to gain by improving reference documentation. After all, reference documentation is just one of the many things that can influence software developer productivity and changing the existing documentation tools and methods is likely to be quite costly. However, if improved documentation can reduce a software developer's interruption in concentration when he or she has a question, the software developer is more likely to keep his or her "flow" [5, 6]. Preventing or minimizing interruptions has a highly leveraged benefit. When software developers lose their flow, it can take them 15 minutes or more to regain it [7]. Eliminating a one-minute, documentation-induced distraction through improved documentation could save 15 minutes of lost software-developer productivity and provide a more satisfying, and more profitable, development experience. Over the course of a software project, such "minutes" add up.

The only way to know for sure if one document design is better than another and if a design can reduce interruptions in concentration is to test the designs empirically. Understanding how software developers use API documentation is also an essential component of improving their reading and learning experience. The tool and method described in this paper intend to do both—empirically measure task performance and collect data to help understand how software developers learn an API. Understanding how different document design elements and styles influence how software developers learn will make it easier for technical writers and documentation tool developers to focus on the aspects that improve the experience for software developers and not waste time on the aspects that do not.

Research Question

The research question that motivates the tools and method described in in this paper concerns whether the design of API reference documentation affects a software developer's performance when he or she refers to the documentation while writing software. Researching such a question requires the analysis of many variables and conditions that make such research time-consuming and costly. Therefore, the first problem to solve is how to lower the cost and complexity of the research. The tools and method described in this paper apply remote user assessment tools and techniques to reduce the cost of this research and address the elements of a problem that has, so far, been difficult and expensive to study.

The research question consists of three elements: the programming task (a constant); the documentation design (the independent variable); and the software developer's task performance (the dependent variable). Answering the overarching research question requires tools and methods that accommodate all of these elements.

Literature Review

The elements of the research question, the programming task, the documentation design, and the task performance, involve these four general areas: learning theory (specifically, adult software developers learning in an online environment), online document-design principles, API design and usability research principles, and Internet-based user-experience research. Learning theory provides the framework to study how a reader finds and consumes information to complete a task and online document-design principles describe how to present information to a reader. API usability literature describes the object used in the tasks and described by the documentation being studied. It also provides information about how to research API usage. The literature on Internet-based user-experience research discusses the means by which we can collect user-assessment data remotely to answer the research question.

I. Learning Theories

The learning theories inform how to model software developers' thinking as they research the information needed to complete programming tasks—that is, use the documentation. Rouet's [8] TRACE model of document processing describes a task-oriented model of document search and information retrieval. Rouet's model is a more detailed version of Wright's [9, 10] document interaction model: searching, understanding, and applying, as referenced by Nielsen [11].

Clarke [12, 13] characterized APIs and API users using 12 cognitive dimensions, one of which is learning style. He identified three different learning styles that software developers use when they learn an API. The learning style applied by the user should match that supported by the API design [12]. Likewise, the task design in a study of an API's usability should consider the learning style of the target audience and the API [14]. If the learning style of an API does not match the learning style applied by the user, one could expect to see poorer task performance.

II. Document Design

Redish [15] describes some of the key design elements of online document design for the reader who "skims and scans." Redish lists elements such as home pages and pathway pages and she discusses how much content to include on a page. Tidwell [16] also describes navigation and presentation patterns that can apply to online documentation. Robillard [2, 3] identified elements of API design that help make APIs easier to learn. A survey of 35 sets of API documentation reviewed the frequency of the more universal high-level design elements described by Redish and the content described by Robillard and Nykaza, et al. [17]. It found that these elements were not present in all of the API documentation sets studied, indicating they might not be as universal as they should be.

III. API Usability Research

API usability research informs several aspects of this project. First, it characterizes the nature of the object the participant uses to perform the task. Clarke [13] used 12 cognitive dimensions to characterize the usability of an API. Characterizing and understanding the usability of the API being used in an experiment makes it possible to evaluate how that might influence the task performance. The second aspect of API usability research that informs this study is how other researchers have studied APIs. While API usability literature describes in-person, moderated studies [13, 14, 18], some of the techniques described in the literature also apply to remote, unmoderated studies. Specifically, for example, we can apply the API usability literature on how to select and design tasks for moderated, in-person API studies to task design for unmoderated, remote user assessment studies. The unmoderated nature of an online, remote user assessment, however, complicates the study somewhat. Clarke [14] describes how the tasks should be designed and written to the persona for which the API is designed. It is difficult for a remote study to screen for participants who match the intended persona; however, such a capability is important.

IV. Internet-Based Research Methods

The literature about studying API use and usability describes laboratory-based, in-person, moderated user studies [13, 14, 18]. While that literature provides valuable insight, such research methods do not scale to a point where very many document designs could be studied in a reasonable period of time or for a reasonable cost. Because there are so many different design combinations and programming situations that could be studied, finding a way to crowd source the research becomes an attractive option. Internet-based research methods inform the methods and tools that can facilitate such an option.

Internet-based research methods provide the ability to collect data from participants who are working in their natural environment and take advantage of a larger sample, however, some of the qualitative details of a participant's environment might not be recorded [19, 20].

STUDY GOAL

Investigating the research question of whether the design of API reference documentation affects a software developer's performance when he or she must refer to the documentation while writing software requires multiple tests of multiple designs, potentially with multiple audiences. While studying these variations in a laboratory setting would provide a rich set of qualitative and quantitative data, the time and expense required to study more than just a few design variables is prohibitive for even a well-funded organization, let alone a small shop or individual technical writer. Designing the experiments to use a remote, Internet-based tool makes this research more practical, accessible, and scalable to the degree necessary to identify the best practices empirically.

Therefore, the ultimate goal of this research is to create a remote user-assessment tool that can accommodate a large number of participants, test a large number of design variations, and produce data that researchers and technical writers can interpret easily. Ideally, the tool would appear familiar to participants and give them the affordances to which they are accustomed, such as an integrated development environment (IDE) that supports software development and multiple browser windows in which to look up information as they program. However, modern browsers and IDEs are complex pieces of software, so before engaging in the potentially time-consuming effort of modifying or adapting existing tools, I designed a simple, HTML-based prototype to test the tool, study concept and methodology, and the data collection and data processing methods to identify the requirements of a more complete research and analysis tool.

METHOD

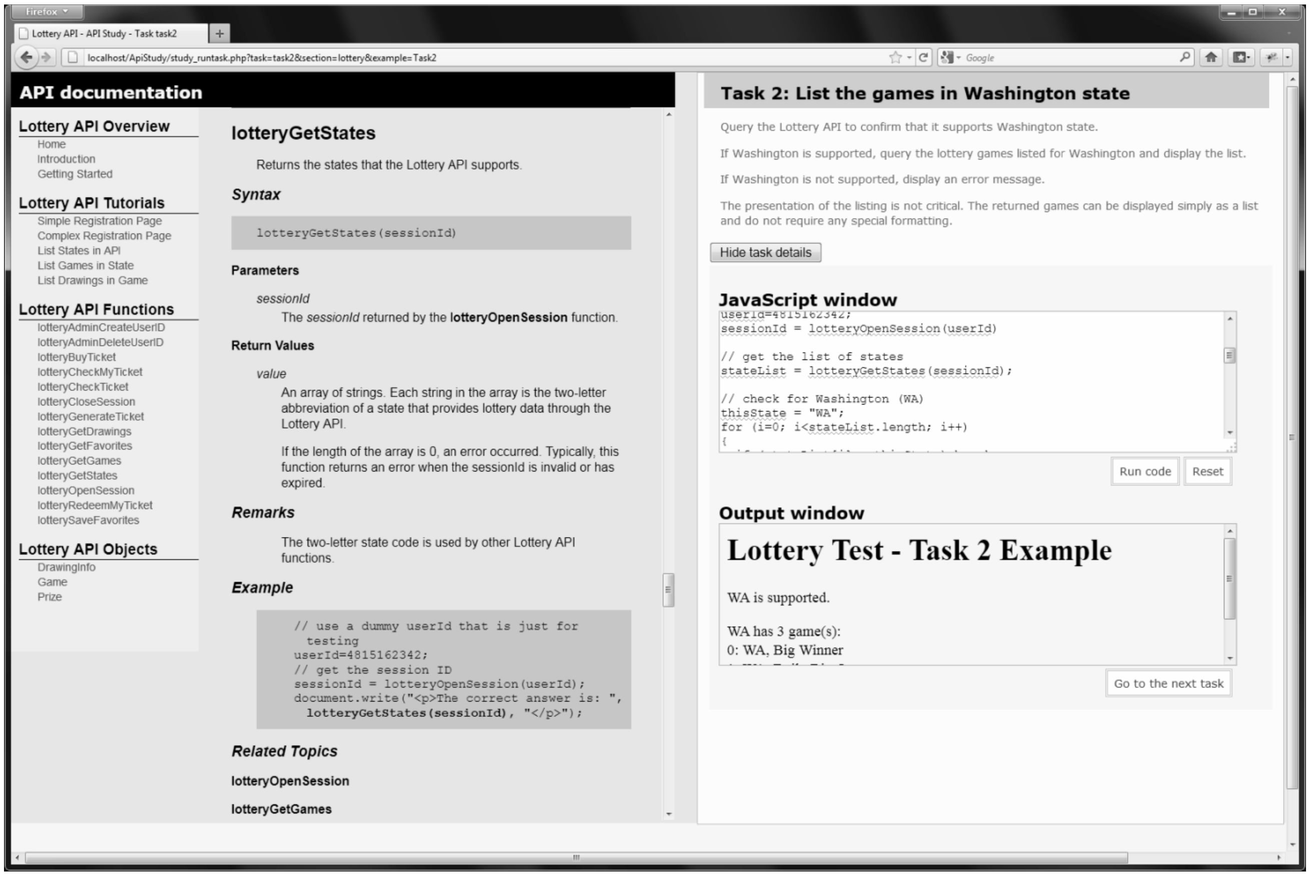
Based on the available literature and the requirements of the experimental design, I built a prototype of a tool that could remotely test programming task performance. I used this tool in a trial study to collect data similar to the data necessary to answer the research question. In this section, I describe the tool, the experiment, and the study used to test the prototype.

I. Tool Prototype Design

The prototype runs in a web browser as a web application to facilitate easy modification and to be compatible with WebLabUX, the Internet-based user-assessment tool. The prototype presents the documentation being studied in the left half of the screen and task-related windows in the right half of the screen. A window at the top of the task-related half of the screen contains the programming task description. Below the task description is the JavaScript window into which participants enter the program code to accomplish the task. Below that is the output window that displays the results of their program code when the participant clicks the "Run code" button. Figure 1 is a screen shot of the prototype as seen by a participant.

The prototype runs in the context of a WebLabUX study session. In the course of a remote user-assessment study, WebLabUX collects such data as the pages visited, the time a participant visits the pages, and the program code that the participant entered. All data collected by WebLabUX contains the time they were recorded, from which WebLabUX can compute task performance data. WebLabUX selects the documentation design seen by a participant by randomly selecting the cascading style sheet (CSS) for that design variation when the participant starts a study session and it uses that design throughout the participant's session. This approach requires having documentation that can be varied by using different CSS files.

To test the prototype, I devised an experiment that tests a specific instance of the research question. It comprised three programming tasks and four document-design variations (two variations of two different document design elements). I chose the design variations for this study based on the high-level design elements surveyed in [17]. The API used in the study came from a small JavaScript library and its documentation that I wrote specifically for this study. Writing a new API guaranteed the novelty of the API so that prior knowledge of the API would not be a factor in the task performance and it let me design and implement the API and the documentation so that it would be compatible with the prototype and WebLabUX.


Lottery API - API Study - Task task2

localhost/ApiStudy/study_runtask.php?task=task2&section=lottery&example=Task2

API documentation

Lottery API Overview
- Home
- Introduction
- Getting Started

Lottery API Tutorials
- Simple Registration Page
- Complex Registration Page
- List States in API
- List Games in State
- List Drawings in Game

Lottery API Functions
- lotteryAdminCreateUserID
- lotteryAdminDeleteUserID
- lotteryBuyTicket
- lotteryCheckMyTicket
- lotteryCheckTicket
- lotteryCloseSession
- lotteryGenerateTicket
- lotteryGetDrawings
- lotteryGetFavorites
- lotteryGetGames
- lotteryGetStates
- lotteryOpenSession
- lotteryRedeemMyTicket
- lotterySaveFavorites

Lottery API Objects
- DrawingInfo
- Game
- Prize

lotteryGetStates

Returns the states that the Lottery API supports.

Syntax

```
lotteryGetStates(sessionId)
```

Parameters

sessionId
The *sessionId* returned by the **lotteryOpenSession** function.

Return Values

value
An array of strings. Each string in the array is the two-letter abbreviation of a state that provides lottery data through the Lottery API.

If the length of the array is 0, an error occurred. Typically, this function returns an error when the sessionId is invalid or has expired.

Remarks

The two-letter state code is used by other Lottery API functions.

Example

```
// use a dummy userId that is just for
  testing
userId=4815162342;
// get the session ID
sessionId = lotteryOpenSession(userId);
document.write("<p>The correct answer is: ",
  lotteryGetStates(sessionId), "</p>");
```

Related Topics

lotteryOpenSession

lotteryGetGames

Task 2: List the games in Washington state

Query the Lottery API to confirm that it supports Washington state.

If Washington is supported, query the lottery games listed for Washington and display the list.

If Washington is not supported, display an error message.

The presentation of the listing is not critical. The returned games can be displayed simply as a list and do not require any special formatting.

JavaScript window

```
userId=4815162342;
sessionId = lotteryOpenSession(userId)

// get the list of states
stateList = lotteryGetStates(sessionId);

// check for Washington (WA)
thisState = "WA";
for (i=0; i<stateList.length; i++)
{
```

Output window

Lottery Test - Task 2 Example

WA is supported.

WA has 3 game(s):
0: WA, Big Winner

FIGURE 1. SCREENSHOT OF PROTOTYPE

II. Data Collection

Deploying the study over the Internet should make it possible to test many different participants and collect data sets that are large enough to show statistically significant differences in task performance, if any differences are present. Unfortunately, the upgraded data collection and reporting features of the WebLabUX were not complete for this test, which meant that I had to analyze the data collected in this test manually. The results from this study, however, will inform the upgrade efforts of WebLabUX.

The raw data collected by WebLabUX during this test include:

- Study condition—which document design did the participant use
- Pre-study questionnaire data
- Post-study questionnaire data
- The documentation topics visited and when they were visited
- Copies of the program code written by the participants, which were saved each time they rendered their program in the output window
- When the participants entered the editing window to write program code, and when they entered the documentation window to use the documentation

III. Study Session Description

An individual study consists of the questionnaires, programming tasks, and an API with its documentation. A study session appears to the participant as the following:

1. A welcome screen with an introduction and the consent form.
2. A pre-task questionnaire that collects demographic data, including three programming questions to assess programming skill.
3. One practice task and three programming tasks.
4. A post-task questionnaire that collects satisfaction and confidence data.
5. A thank you screen.

WebLabUX manages the questionnaires, their data, and records the web-site interactions.

IV. Study Population

I used a convenience sample for this study because the goal of the study was to test the tool and not evaluate the participants or any population they might represent. I sent invitations to university students and professional software developers to collect a range of responses that exercised the tool in a variety of environments.

FINDINGS

I sent invitations by e-mail to 531 people, 12 people responded and started the study; however, only four completed it. This represents a 2.3% response rate, a 0.8% completion rate, and a 66.7% dropout rate. The participants who completed the study took an average of 17.45 minutes to complete it; however, two participants took an average of 13.5 minutes and the other two took an average of 21.4 minutes.

Of the eight participants who dropped out, four left while in the first task, and four left in the second task, but after completing the first task. The average time spent in the study by participants who did not finish was 3.87 minutes; however, three participants spent more than 10 minutes in the study before they left.

DISCUSSION

In this section, I discuss my experience with the prototype in the context of a study. The convenience sample method and low response rate prevents drawing any conclusions about the document design variations selected for the study; however, the data collected by the tool from the 12 participants who responded provide sufficient information about the changes to include in future versions of the tool.

I. Participant Response

I sent invitations by e-mail to 531 people, expecting a 10% completion rate. This expectation came from a previous study that I sent to the same group a year earlier, which yielded a 10% completion rate. In this study, the response rate was lower than expected and the dropout rate was higher. I list some of the possible reasons for this later in this discussion, but the key finding here, is that 10% might represent the high end of the range of response rates for this type of study. After addressing some of the mitigating factors of this study, a more reasonable expectation for future studies might be to expect 10% of those invited to respond and half of those (5% of the people invited) to result in a session that yields valid data.

Because the goal of this study was to test the prototype, a large response is not required. In fact, because the data required manual processing and analysis, a larger response might have provided too much data to analyze. However, to improve the response rate, future studies could apply some of the techniques Dillman [21] describes to improve the response rate of Internet-based studies. For example, in this study, I sent only one invitation to each of person. Dillman, instead, suggests sending several messages and reminders and using increasingly compelling and persuasive text in the invitations. He also suggests offering a gratuity or the chance for a gratuity to help improve the response rate. This study, however, was limited by having only one approved invitation message.

An improved task environment could help reduce the dropout rate. For example, a more helpful programming environment might have reduced some of the frustrations that one participant experienced trying to finish a task. In addition, future studies could include fewer tasks. If each task takes about 5-10 minutes to complete, a study with three tasks could take 30 minutes or more to complete. Designing the experiments such that they require only 10-15 minutes, instead of 30 minutes would help reduce the dropout rate.

It is important to select the sample carefully. Clark [14] mentions matching the tasks to the persona of the target audience. I designed the tasks primarily for JavaScript programmers; however, I did not pre-screen the audience who received the invitations for any programming-language preference. In this study, only one of the participants reported JavaScript as the programming language they used the most, while six of the participants reported programming in C# as the programming language they used the most. The C# programmers in the study, however, reported and demonstrated a high degree of competence using JavaScript in the pre-task questionnaire. Curiously, the C# programmers spent less time in the study, yet they interacted with the study more than the non-C# programmers did. However, because two-thirds of the participants reported a high degree of familiarity with JavaScript and answered both JavaScript test questions in the pre-task questionnaire correctly, it is possible that the C# programmers lost patience with the study, but it is unlikely that a lack of JavaScript proficiency was their reason for dropping out early.

II. Task Experience

The prototype provided an adequate task experience for the participants, but as mentioned earlier, it will be necessary in future studies to use an interface with which the participants are familiar and that offers the affordances and programming assistance with which they are accustomed. One shortcoming in the study design that the prototype testing identified was that the prototype did not offer a way to leave the programming tasks early in a way that took them to the exit survey. Participants who left the study early might have answered the short exit survey and said why they left early if they had been given the opportunity. Not allowing for this in the prototype caused this information to be lost. A future implementation of the tool should let the participant leave early and still offer them the opportunity to complete the exit survey.

III. Document Experience

I wrote the documentation used in the study specifically for this test and for the WebLabUX environment. The current implementation of WebLabUX manages variations by controlling CSS styles, which are defined in separate CSS files. This requirement limits the extent of design variations that WebLabUX can test to only those that can be manipulated by changing a style definition. The documentation in this study accommodated this requirement; however, this could prevent testing some design variations in existing documentation.

The development environment provided by the prototype did not support the inline programming assistance that a developer would usually have when they use an IDE. As a result, the participants did not have access to a source of documentation that they would commonly find in modern development tools. I also observed in early pilot testing of the prototype that the participants opened additional browser windows for help with JavaScript programming. WebLabUX does not track these additional windows, so it is possible that the participants spent some of the time in was spent accessing other resources besides the documentation provided by the study.

IV. Data-Collection Experience

The data collection facilities of WebLabUX were still in development when I conducted this study to test the prototype. The prototype required additional extensions to the standard data logging that WebLabUX provides. These factors made calculating the performance data a manual (and tedious) task. The low participation rate still provided enough samples to develop future data-analysis methods and kept the dataset manageable for manual calculations, even if it prevented generalizing any observations from the data.

Because the prototype did not clearly delineate the task boundaries in the data that WebLabUX recorded, having multiple tasks in the study complicated data analysis. The data required additional post-processing before they could be interpreted. Post processing the data is not necessarily bad, but it complicates the scaling and automation of the data analysis and places additional requirements or limitations on the study.

At a more detailed level, the design of the study did not make it possible to distinguish the time the participant spent reading the task instructions from the time they spent solving the task. In an easy task, the time spent reading the task instructions could be a considerable percentage of the total time spent in the task. The distinction between reading and performing the task will need to be clear in the next version of the tool.

The document path taken by the participants is visible in the session log; however, the prototype injects other data into the session log requiring the separate data elements to be extracted after the study. In this study, the prototype sent the following datasets to WebLabUX:

- The task performance data – the start and end time of each task.
- The document path data – the sequence of pages visited by the participant while they perform the task.
- The interaction sequence data – the sequence of state changes that occur when the participant goes from editing in the task window to reading in the document window and when he or she goes from the document window back to the task window.
- The source code data – the program code created by the participant in the process of completing the task.

In this study, the prototype interleaved all of the datasets in a single session log, so the session log required post-processing to separate them. For this study, I separated the elements manually; however, in a larger-scale study, these elements must be clearly identifiable so they can be separated automatically.

The most complex data to analyze systematically is the sample code that the WebLabUX collected and recorded each time participants render the source code they entered in the task window. For this study, I examined the program code manually; however, some of the options that future studies could pursue include:

- Evaluating the correctness of the last solution—does the last sample of program code submitted by the participant produce the results described in the task instructions?
- Tabulating the number of submission attempts per task—how many tries did participants make before they felt they had completed the task?
- Evaluating how much the program code changed between submissions—how much did participants change the code before they submitted it again? This could be measured in the number of characters that changed from one sample to the next.

The last two items could provide insight into the participants' programming styles. For example, do they make frequent, small changes or infrequent, large changes?

Conclusions and Future Work

This study tested a method and identified the requirements of a tool that will make it easier to design and deploy studies that evaluate document designs and answer the research question empirically. This study found the following about the method and the tools.

I. Method

The method is sound and supported by existing literature; however, the technical challenges uncovered by the study, and described in the following sections, must be addressed. Some of the weaknesses of conducting this research remotely include the inability to collect more data about the participants' environments and not having the ability to make specific observations when the participant encounters a problem. Both of these could provide valuable input into making design decisions; however, valid conclusions and design decisions can still be made with the data provided by the tool. The data collected from the larger sample that an Internet-based study can access will include the influence of environmental variations, even if the underlying cause of such variations is unknown.

This, however, brings up the concern that the task performance data collected by this method will include variations that result from factors other than the independent variables (document designs) being studied. Delays that result from, in-situ, participant distractions and programming-related delays due to typographical errors and missing punctuation cannot be distinguished easily from problems with the documentation or design variation. On the one hand, that is a valid concern in that such distractions and external influences taint the purity of the performance measurements. On the other hand, those are the very distractions we are hoping to encounter by using remote user assessment. The ideal design identified by these experiments is the one that works best in the users' environments. It is much more important that a design works best in the users' environments than it does in a usability lab. The best design is the one that works best amongst all the distractions a real user encounters, whatever they might be. That being said, and as mentioned above, this method by itself does not provide much information about what those distractions might be. When it becomes necessary to identify specific distractions, another method might provide data that are more useful.

II. Task Interface

The task interface used by the prototype in this study provided a basic development experience for the participant. However, it provided very little, if any, of the support that most software developers now expect in a development environment. I designed the tasks to be simple enough to complete with just a small procedure to minimize the impact of this; however, the programming language used, JavaScript, is sensitive to minor typographical errors such as missed punctuation or inconsistent letter casing. As such, it is easy for a software developer to spend a lot of time troubleshooting non task-related problems in the prototype's JavaScript window. Typically, a software developer would use a development environment that identifies simple errors as they write a program so they can fix them quickly and almost without thinking. The coding window of the prototype used in this study, however, provided no such help, leaving open the question of, "did a participant take the time they did because they could not find a topic in the documentation or because the program had an error they could not find easily?" Analyzing the source code submissions provides some insight, but this is difficult to detect automatically. If such errors could be identified, those samples could be excluded from the task-performance data analysis or the time could be adjusted to account for the non-task-related activities.

Modern development environments typically include some sort of context-sensitive, inline help or information about commonly used programming functions and APIs. The method used in this study did not provide this type of assistance nor was it instrumented to measure it. It would seem that having this type of assistance could speed up task completion, so because it is common, useful, and beneficial, a study of programming task performance should account for its influence. A more complete test tool would support this and measure the usage of this type of help as a part of a documentation design study.

III. Document Interface

The API and its corresponding documentation was developed and coded specifically to test the prototype in this study. Ideally, it would be possible to study any documentation set while requiring little, if any, modification of the documentation. Based the results of this study, I am investigating several options. The least intrusive option for the technical writer would be to incorporate the necessary instrumentation and study parameters into the content management system used to create the documentation. Incorporating this instrumentation into the content management system would allow a writer to author his or her content as they normally would, while giving little consideration to the mechanics of running a study. Document variations could be designed as themes, for example, in the content management system and the study tool would manage and monitor the theme seen by any individual user. After authoring the content, the writer could then study the effect of different themes (document presentation styles) on task performance using the user research tool.

A more flexible option might be to provide a means by which web pages from any content management system could be instrumented and monitored. Creating a link that can be included in an existing documentation set, in a similar fashion to how web metrics are currently collected, enables data to be collected from any web-based documentation set. However, the documentation would also need to be modified to accommodate the different study conditions, for example, through the selection of style sheets. This option requires more involvement by the writer than the previous method, but offers more flexibility in terms of what type of content can be studied.

From the user's perspective, the document interface used in a study should use the same web browser or browsers that they are accustomed to using normally to minimize any effects caused by the study tool.

IV. Future Work

Using the results of this study, I will work with the University of Washington's Internet-Based User Experience Lab to review how they can make WebLabUX more versatile. We will identify how to modify WebLabUX to accommodate a wider range of experiments and how to make it extensible such that it can accommodate special data collection and analysis.

I am investigating the use of browser extensions to collect study data directly from the web browser. This will provide the user/participant with a natural interface through which to access the documentation and provide information about their browsing whether they access a page that is instrumented for study or not.

It could also be possible to generalize this type of analysis to other reading to learn to do tasks that can be accomplished in a browser interface such as in a training scenario where people are learning how to use a new browser-based application.

Acknowledgements

I want to extend my thanks to Dr. Jan Spyridakis, Elisabeth Cuddihy, the University of Washington's Internet-Based User Experience Lab, and the researchers who are helping develop the next version of WebLabUX. Without their help and collaboration, I could not have accomplished this research.

References

[1] T. C. Lethbridge, J. Singer and A. Forward, "How software engineers use documentation: The state of the practice," *Software, IEEE*, pp. 35-39, November-December 2003.

[2] M. P. Robillard, "What Makes APIs Hard to Learn? Answers from Developers," *Software, IEEE*, vol. 26, no. 6, pp. 27-34, November/December 2009.

[3] M. P. Robillard and R. DeLine, "A field study of API learning obstacles," *Empirical Software Engineering*, pp. 703-732, 2011.

[4] J. Nykaza *et al*, "What Programmers Really Want: Results of a Needs Assessment for SDK Documentation," in *Proceedings of the 20th Annual International Conference on Computer Documentation*, Toronto, Ontario, Canada, 2002.

[5] T. DeMarco and T. R. Lister, *Peopleware: Productive projects and teams*, New York: Dorset House Pub. Co., 1987.

[6] J. Spolsky, *Joel on software: And on diverse and occasionally related matters that will prove of interest to sofware developers, designers, and managers, and to those who, whether by good fortune or ill luck, work with them in some capacity*, Berkeley: Apress, 2004.

[7] T. DeMarco and T. P. Lister, "Programmer performance and the effects of the workplace," in *Proceedings of the 8th international conference on Software engineering*, London, England, 1985.

[8] J. F. Rouet, *The Skills of Document Use: From Text Comprehension to Web-Based Learning,* Mahweh, NJ: Lawrence Erlbaum Associates, 2006.

[9] P. Wright, "Designing and evaluating documentation for I.T. users," in *Human Factors for Informatics Usability*, B. Shackel and S. Richardson, Eds., Cambridge, Cambridge University Press, 1991, pp. 343-358.

[10] P. Wright, "Manual dexterity: A user-oriented approach to creating computer documentation," in *Proc. ACM CHI '83 Conf.*, Boston, MA, 1983.

[11] J. Nielsen, *Usability Engineering*, Boston, MA: Academic Press, 1993.

[12] S. Clarke, "Using the cognitive dimensions, continued - learning style," 24 11 2003. [Online]. Available: http://blogs.msdn.com/stevencl/archive/2003/11/24/57079.aspx. [Accessed 2 2 2010].

[13] S. Clarke, "Measuring API Usability," *Dr. Dobbs Journal Special Windows/.NET Supplement*, pp. S6-S9, May 2004.

[14] S. Clarke, "HOWTO: Design and run an API usability study," 29 March 2005. [Online]. [Accessed 10 May 2008].

[15] J. Redish, *Letting Go of the Words: Writing Web Content that Works*, San Francisco, CA: Morgan Kaufmann Publishers, 2007.

[16] J. Tidwell, *Designing Interfaces*, Sebastopol, CA: O'Reily Media, Inc., 2006.

[17] R. Watson, "Development and application of a heuristic to assess trends in API documentation," Manuscript submitted for publishing in *Proceedings of ACM SIGDOC Conference*, Seattle, 2012.

[18] S. G. McLellan *et al*, "Building More Useable APIs," *Software, IEEE*, pp. 78-86, May/June 1998.

[19] J. Spyridaki *et al*, "Internet-Based Research: Providing a Foundation for Web-Design Guidelines," *IEEE Transactions on Professional Communication*, vol. 48, no. 3, pp. 242-260, September 2005.

[20] C. Wei *et al*, "Conducting Usability Research through the Internet: Testing Users via the WWW," in *Proceedings of the Usability Professional Association*, Montreal, 2005.

[21] D. A. Dillman *et al, Internet, mail, and mixed-mode surveys: the tailored design method, 3rd. ed.*, Hoboken, New Jersey: John Wiley & Sons Inc., 2009.

About the Author

Bob Watson developed software professionally for 17 year. After writing software for many products and designing many APIs, he stopped writing software and started writing about software as a programming writer at Microsoft. As a programming writer, he has been writing about software for software developers for almost 10 years. He holds a Master of Science degree in Human Centered Design and Engineering from the University of Washington with a focus on user-centered design and is currently studying for his PhD in Human Centered Design and Engineering and Global Technical & Communication Management.