Usable Results from the Field of API Usability

A Systematic Mapping and Further Analysis

Chris Burns, Jennifer Ferreira, Theodore D. Hellmann, Frank Maurer
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada
{chris.burns, jen.ferreira, tdhellma, frank.maurer}@ucalgary.ca

***Abstract*—Modern software development often involves the use of complex, reusable components called Application Programming Interfaces (APIs). Developers use APIs to complete tasks they could not otherwise accomplish in a reasonable time. These components are now vital to mainstream software development. But as APIs have become more important, understanding how to make them more usable is becoming a significant research question. To assess the current state of research in the field, we conducted a systematic mapping. A total of 28 papers were reviewed and categorized based on their research type and on the evaluation method employed by its authors. We extended the analysis of a subset of the papers we reviewed beyond the usual limits of a systematic map in order to more closely examine details of their evaluations – such as their structure and validity – and to summarize their recommendations. Based on these results, common problems in the field are discussed and future research directions are suggested.**

Keywords-API usability; systematic map; systematic review; meta-analysis; application programming interface

I. INTRODUCTION

In modern software development projects, developers are spending more and more time programming through Application Programming Interfaces (APIs). Rather than developing functionality from scratch, developers find and reuse an API that meets their needs. Nearly all development work is done through the use of countless APIs – to interact with the user interface, to communicate with the database, to process data, and so on. In a blog post published in 2010[1] Mike Taylor described modern programming as "mostly pasting libraries together" and quoted Donald Knuth, who said that programming today is "just plugging in magic incantations – combine somebody else's software and start it up."[2]. This article suggests that a fundamental shift has occurred and that modern software development is primarily *API-driven development*. However, an API will slow down development and reduce developer productivity if it is designed poorly, is difficult to learn, is undocumented, or is generally difficult to use. Based on this, understanding how to make APIs usable is an important research topic with a direct impact on developer practice [1].

Because of this fundamental shift in how development work is done, it would be reasonable to expect that research has changed as well. API Usability is an established research topic and many papers on the topic have been published, so it would be useful to understand the current state of the art. To accomplish this, we performed a meta-analysis to characterize the published API Usability literature. We started our investigation by creating a systematic map to get a high-level understanding of this field. Based on this map, we performed a deeper investigation into the ways in which researchers have evaluated the usability of APIs and their findings. This deeper analysis, where papers are read completely, is in the style of a systematic literature review. Our work is thus a combination of both similar to work by Bastos et al. [2] This work should be considered a pilot study and a precursor to a more detailed systematic literature review.

II. RESEARCH METHOD AND CONDUCT

This study is based on the guidelines provided by Petersen et al. [3] and Kitchenham and Charters [4] as well as on previous work done by Bastos et al. [2]. This section provides details on how we conducted each step of our investigation.

A. Research Questions

Any systematic analysis begins with a series of motivating questions which direct the investigation [3,4]. The research questions which motivate this study are as follows:

1) **RQ 1.** How have APIs been evaluated?
2) **RQ 2.** What kinds of recommendations have been made to improve API usability?

B. Data Sources and Search Queries

Search strings were created to query the IEEE Xplore[3] and SciVerse Scopus[4] databases. Xplore was chosen because the IEEE hosts many conferences related to computer science and software engineering. Scopus is also an important source because it includes the results of other major research databases (for example, the ACM Digital Library and Springerlink). The search string used on both of these databases was "(API OR Framework OR Toolkit OR Library) AND Usability." The keywords used in the search string are commonly used synonyms that are interchangeably used in the API literature.

[1] M. Taylor. (2010, March) The Reinvigorated Programmer: Whatever happened to programming? http://reprog.wordpress.com/2010/03/03/whatever-happened-to-programming/
[2] P. Siebel, Coders at Work: Apress, 2009.
[3] http://ieeexplore.ieee.org
[4] http://www.scopus.com

C. Inclusion and Exclusion Criteria

Using the search queries detailed above, 39 papers were returned. Inclusion and exclusion criteria were applied to the titles and abstracts returned by the queries. The introductions were read when it was not clear from the titles and abstracts whether the papers were addressing API usability to eliminate those which were outside the scope of the review. After the inclusion and exclusion criteria were applied, 28 papers remained.

The inclusion criteria are as follows:

1) *English language:* No results were non-English, therefore, all the results were eligible for inclusion.
2) *Availability of full paper including abstract:* If the search results were linked with a full paper that included an abstract, they were eligible for inclusion.

The exclusion criteria are as follows:

1) *No mention of APIs*: To be included, a paper needed to have some discussion of APIs. If the paper discussed the usability of an interface that was not programmer-facing it was not included. The papers excluded by this rule were often focused on usability in a more general sense.
2) *APIs mentioned indirectly*: Some papers discussed APIs but were not focused on them as a goal of the research. This rule was carefully applied; a paper was only rejected if APIs were incidental to the research of the paper.
3) *Design issues outside the scope of APIs*: Some papers involved claims about design that were not limited to or targeted at APIs. Papers that discussed programming practices for software in general were not included.

Of the 28 chosen papers many were published in software engineering and human-computer interaction conferences.

D. Keywording and Categorization

In order to organize the collected papers for categorization, an initial review was conducted to assign keywords to all the papers based on their abstracts. The keywords were iteratively refined as each paper was considered. The paper set was categorized in two ways: by type of research and by type of evaluation.

Categories were designed such that papers could only fit one. The categories describing research type are as follows:

1) *Design Paper:* Focuses on some aspect of API design (such as architecture, design patterns, or coding practices). [5,6,7,8,9,10,11]
2) *Framework Paper:* Presents a methodology or framework for determining the usability of an API. Provides a procedure which could be used to evaluate any API. [1,12,13,14,15,16,17,18]
3) *Tool Paper:* Presents a tool that would improve the usability of an API by assisting developers using the API (including extensions to an IDE and web applications). [19,20,21,22,23,24]
4) *Examination Paper:* Focuses on a specific subset of APIs and draws conclusions that should be applicable to all APIs. [25,26,27,28,29,30,31]

E. Analysis of Evaluations

After the papers had been categorized, it was possible to review them more closely and assess their evaluations. Specifically we considered the *Tool Papers* and the *Design Papers.* In order to accomplish a review of these, it was necessary to fully read each paper rather than just the abstracts or introductions. In this sense analysis departs from the usual procedure of a systematic map and becomes a systematic review with a strongly restricted scope. The user studies are compared based on their common attributes. These include the number of participants in the study, the number of tasks completed, the length of time allowed for the tasks, and whether a control and experimental group were used.

F. Analysis of Recommendations

The recommendation made by a paper –what the authors advised API designers to do – was potentially the most important information gathered through this analysis. This area was problematic to categorize because many of the recommendations were unique and specific to the paper and approach used. To deal with this difficulty, the recommendations made in each reviewed paper were succinctly summarized and these summaries collected and organized into categories.

III. Results

Our investigation produced a detailed analysis of the evaluations and the recommendations made by paper authors as well as an overview of the methodologies used to evaluate APIs.

A. Analysis of Evaluations

In considering the evaluations used in the paper set we focused on the *Tool Papers* and *Design Papers*. Out of 13 papers in these two categories only 5 had *User Study* evaluations and the rest were considered to have no evaluation at all. For every user study, the number of participants, the number of tasks, the length of those tasks, and the use of control and experimental groups was determined.

Some preliminary statements can be made based on the information displayed in Table I. In forty percent of the evaluated user studies, the distinction between control and experimental factors was not made or was not made clear. The small number of participants in some of the user studies is also a cause for concern. Additionally, it is difficult to evaluate many aspects of an API – such as its ease of incorporation, flexibility in various use cases, and learnability –though short user studies. As the field of API usability evolves, it will be important to expand on the evaluation methods and consider new evaluation approaches. Some of these are considered in the Future Work section.

TABLE I. ATTRIBUTES OF USER STUDIES

Paper	Participants	Tasks	Task Length	Control & Experimental
1	26	5	15 min	Unspecified
2	12	5	Not Limited	Yes
3	30	6	30 min	Yes
4	7	3	Dependent	Yes
5	2	1	Unspecified	Unspecified

B. Analysis of Recommendations

Each paper that had some empirical justification for its recommendations was reviewed for the recommendations regarding APIs:

1) *Design Recommendations:* simple forms of object creation are preferable; constructors should be used instead of factories and should not force developers to set all parameters; simple architecture is better; avoid overuse of inheritance and configuration files. [6,10,9]

2) *Documentation Recommendations:* provide useful examples; documentation should integrate with the IDE and should inform developers of metadata regarding an API's classes, objects, and methods; unit tests could be used as a form of documentation instead of usage examples. [30,29,19]

3) *Methodology Recommendations:* API usability should be evaluated using actual developers; developers should "think aloud" to explain what they are doing and why during user studies; a facilitator can walk participants through code to draw out participants' assumptions about an API. [15,28,13]

C. Summary of Methodologies

Some of the methodologies for evaluating an API were discussed in the recommendation section, but only those with some justification were considered. A wider set of the methodologies used is found in Table II.

IV. DISCUSSION

This meta-analysis set out to answer a set of research questions. The results of this review, threats to its validity, and some problems with existing research are discussed in this section.

A. Research Questions

RQ1 asked what methods could be used to evaluate an API. We found that *User Studies* were the typical method and that some discussion of was presented. A discussion of the *Methodologies* that researchers proposed for evaluating a given API was also considered. Of these papers, the recommended methodologies borrow heavily from the field of human-computer interaction.

RQ2 asked what recommendations were made to developers and this was discussed in section III. In general, the most concrete recommendations dealt with simple design choices such as which object initialization pattern to use.

B. Threats to Validity

There are some clear threats to the validity of the systematic mapping. These come in two forms: subjective decision problems and data collection problems. Since only one researcher was involved in the categorization and elimination of papers, it's possible that some subjective bias could impact the results. To mitigate this issue, inclusion and exclusion criteria and the category definitions were made as straightforward and unambiguous as possible. Since only two data sources were used and relatively simple search strings were employed, it's also possible that some papers of interest were not included.

V. FUTURE WORK

The results of the systematic mapping and further review exposed some interesting trends in evaluations of APIs. A further systematic review of the field and the use of more rigorous evaluation methods would be useful future work.

A. Further Meta-Analysis

A systematic map is designed to produce a high level overview of a research field, but this often comes at the cost of detail and precision. Conducting a full systematic review would produce a fuller characterization of the API Usability field.

B. More Detailed Evaluations

As mentioned in the earlier sections, API Usability evaluations often employ user studies. These studies typically involve a small number of tasks being completed over a relatively short period of time. These tasks are often contrived and involve only a subset of the functionality of the API in question. This approach assumes implicitly that, like user interfaces designed for the general public, an API is usable if these simple tasks can be accomplished quickly by relatively naïve users. However, this sort of analysis will be heavily influenced by a user's experience with similar APIs.

TABLE II. SUMMARY OF METHODOLOGIES IN RESULT SET

Methodology	Papers	Summary
Cognitive Dimensions	2	Dimensions are a series of independent "measures" which can be defined for an API. A standard set called *cognitive dimensions* measures twelve and a modified version measures only four. Some examples of cognitive dimensions are: abstraction level, domain correspondence, and penetrability.
Reviews	2	Reviews involve having a developer work with an API while a usability engineer or facilitator works with them to learn something about the system. The exact procedure for this varies.
Mathematical Approaches	3	These approaches use set notation, ontology arguments, and graph theory to derive a quantitative measure of the usability of an API.
Concept Maps	1	This approach attempts to illustrate usability problems with an API through a loosely-structured diagram.

Evaluations which considered a more detailed evaluation over a longer period of time would be beneficial to the field.

VI. CONCLUSION

This research aimed to characterize the current state of research in the field of API Usability and to investigate several important topics. By systematically mapping existing literature, it was possible to describe the types of research conducted and the methods used to evaluate that research. Further analysis was necessary to focus on two additional questions: what is the validity and quality of the evaluations; and what recommendations does the literature make to API designers. It was possible to indicate areas for future research and describe several forms of future work that could be accomplished with this information. The field of

REFERENCES

[1] S. Clarke, "Measuring API Usability," *Dr. Dobb's Journal*, pp. S6-S9, May 2004.

[2] J. F. Bastos, P. A. Neto, E. S. Almeida, and S. R. Meira, "Adopting software product lines: A systematic mapping study," in *Evaluation & Assessment in Software Engineering, 2010*, 2011, pp. 11-20.

[3] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic Mapping Studies in Software Engineering," in *International Conference on Evaluation and Assessment in Software Engineering*, Bari, Italy, 2008, pp. 71-80.

[4] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering Version 2.3," Keele University, Staffordshire, Technical Report EBSE200701, 2007.

[5] J. Daughtry, "The Style and Substance of API Names," in *IEEE Symposium on Visual Languages and Human-Centric Computing, 2010*, Madrid, Spain, 2010, pp. 259-260.

[6] B. Ellis, J. Stylos, and B.A. Myers, "The Factory Pattern in API Design: A Usability Evaluation," in *IEEE International Conference on Software Engineering, 2007*, Minneapolis, MN, 2007, pp. 302-312.

[7] D. Hou, C. Rupakheti, and H. Hoover, "Documenting and Evaluating Scattered Concerns for Framework Usability: A Case Study," in *Asia-Pacific Software Engineering Conference, 2008*, Beijing, China, 2008, pp. 203-220.

[8] S. Ramkumar, S. Kumar, and R. Shiroor, "Process centric guidance and tools for next generation Network Services API design," in *IEEE International Conference on Internet Multimedia Services Architecture and Application, 2010*, Bangalore, India, 2010, pp. 1-6.

[9] T. Scheller and E. Kuhn, "Measurable Concepts for the Usability of Software Components," in *EUROMICRO Conference on Software Engineering and Advanced Applications, 2011*, Oulu, Finland, 2011, pp. 129-133.

[10] J. Stylos and S. Clarke, "Usability Implications of Requiring Parameters in Objects' Constructors," in *International Conference on Software Engineering, 2007*, Minneapolis, MN, 2007, pp. 529-539.

[11] J. Stylos and B.A. Myers, "Mapping the Space of API Design Decisions," in *IEEE Symposium on Visual Languages and Human-Centric Computing, 2007*, Coeur d'Alene, Idaho, 2007, pp. 50-60.

[12] C. Bore and S. Bore, "Profiling software API usability for consumer electronics," in *IEEE International Conference on Consumer Electronics*, Signapore, 2005, pp. 155-156.

[13] U. Farooq, L. Welicki, and D. Zirkler, "API usability peer reviews: a method for evaluating the usability of application programming interfaces," in *International Conference on Human Factors in Computing Systems, 2010*, Atlanta, Georgia, 2010, pp. 2327-2336.

[14] J. Gerken, H. Jetter, M. Zöllner, M Mader, and H Reiterer, "The concept maps method as a tool to evaluate the usability of APIs," in *International Conference on Human Factors in Computing Systems, 2011*, Vancouver, BC, 2010, pp. 3373-3382.

[15] P. O'Callaghan, "The API walkthrough method: a lightweight method for getting early feedback about an API," in *Evaluation and Usability of Programming Languages and Tools*, Reno, Nevada, 2010, pp. 1-6.

[16] D. Ratiu and J. Juerjens, "Evaluating the Reference and Representation of Domain Concepts in APIs," in *IEEE International Conference on Program Comprehension, 2008*, Amsterdam, Netherlands, 2008, pp. 242-247.

[17] D. Ratiu and J. Jurjens, "The Reality of Libraries," in *European Conference on Software Maintenance and Reengineering, 2007*, Amsterdam, Netherlands, 2007, pp. 307-318.

[18] C.R.B de Souza and D.L.M Bentolila, "Automatic evaluation of API usability using complexity metrics and visualizations," in *IEEE International Conference on Software Engineering, 2009*, Vancouver, BC, 2009, pp. 299-302.

[19] U. Dekel and J.D. Herbsleb, "Improving API documentation usability with knowledge pushing," in *IEEE International Conference on Software Engineering, 2009*, Vancouver, BC, 2009, pp. 320-330.

[20] D.S. Eisenberg, J Stylos, and B.A. Myers, "Apatite: a new interface for exploring APIs," in *International Conference on Human Factors in Computing Systems*, Atlanta, Georgia, 2010, pp. 1331-1334.

[21] D.M. Pletcher and D. Hou, "BCC: Enhancing code completion for better API usability," in *IEEE International Conference on Software Maintenance, 2009*, Edmonton, Canada, 2009, pp. 393-394.

[22] J Stylos, A Faulring, Y Zizhuang , and B A Myers, "Improving API documentation using API usage information," in *IEEE Symposium on Visual Languages and Human-Centric Computing, 2009*, Corvallis, Oregon, 2009, pp. 119-126.

[23] R.B. Watson, "Improving software API usability through text analysis: A case study," in *IEEE Internationa lProfessional Communication Conference, 2009*, Po'okela, Hawaii, 2009, pp. 1-7.

[24] Y.C. Wu, L.W. Mar, and H.C. Jiau, "CoDocent: Support API Usage with Code Example and API Documentation," in *International Conference on Software Engineering Advances, 2010*, Nice, France, 2010, pp. 135-140.

[25] J. Beaton, S. Y. Jeong, Y. Xie, J. Stylos, and B.A. Myers, "Usability challenges for enterprise service-oriented architecture APIs," in *IEEE Symposium on Visual Languages and Human-Centric Computing*, Herrsching am Ammersee, Germany, 2008, pp. 193-196.

[26] D. Hou and L. Li, "Obstacles in Using Frameworks and APIs: An Exploratory Study of Programmers' Newsgroup Discussions," in *IEEE Conference on Program Comprehension, 2011*, Kingston, Ontario, 2011, pp. 91-100.

[27] M. Maleshkova, C. Pedrinaci, and J. Domingue, "Investigating Web APIs on the World Wide Web," in *IEEE European Conference on Web Services, 2010*, Lugano, Switzerland, 2010, pp. 107-114.

[28] S. McLellan, A. Roesler, J. Tempest, and C. Spinuzzi, "Building More Usable APIs," *IEEE Software*, vol. 15, no. 3, pp. 78-86, 1998.

[29] S.M. Nasehi and F. Maurer, "Unit tests as API usage examples," in *IEEE International Conference on Software Maintenance, 2010*, Timisoara, Romania, 2010, pp. 1-10.

[30] M. Robillard, "What Makes APIs Hard to Learn?," *IEEE Software*, vol. 26, no. 6, pp. 27-34, 2009.

[31] J. Stylos et al., "A case study of API redesign for improved usability," in *IEEE Symposium on Visual Languages and Human-Centric Computing, 2008.*, Herrsching am Ammersee, Germany, 2008, pp. 189-192.