2009

API usability: Report on Special Interest Group at CHI

John Daughtry

Umer Farooq

Brad Myers
Carnegie Mellon University

Jeffrey Stylos
Carnegie Mellon University

API Usability: Report on Special Interest Group at CHI

John M. Daughtry
The Pennsylvania State Univ.
daughtry@psu.edu

Umer Farooq
Microsoft Corp.
umfarooq@microsoft.com

Brad A. Myers and Jeffrey Stylos
Carnegie Mellon Univ.
{bam, jsstylos}@cs.cmu.edu

http://www.apiusability.org/
DOI: 10.1145/1543405.1543429
http://doi.acm.org/10.1145/1543405.1543429

Abstract

The 27th annual International Conference on Human Factors in Computing (CHI) convened in Boston, MA (USA) from April 4-9, 2009. Included in this year's technical program was a special interest group (SIG) meeting on API usability. This report summarizes the SIG, emphasizing the primary takeaways, which include a greater understanding of the types of APIs, case studies, and a place to share our multi-disciplinary results.

Keywords: Software Engineering, Human-Computer Interaction, Application Program Interface (API), Psychology of Programming, Empirical Studies of Programmers (ESP), Natural Programming, Software Libraries.

Introduction

The fields of software engineering (SE) and human-computer interaction (HCI) cross paths in many subareas such as requirements elicitation and the evaluation of software systems. In addition, software developers are end-users of programming languages, integrated development environments, and configuration management tools. Thus, the principles and methods of HCI apply to these artifacts just as they apply to the finished software systems that programmers create.

It is clear that the SE community has long considered the software module as a first-order artifact within their research endeavors. Likewise, the HCI community has long held an interest in the larger programming activity (e.g., [6,16,18,19,22,23,24]). However, neither field has devoted significant attention to the usability of Application Program Interfaces (APIs) as a first-order object of study.

Certainly, subareas of software engineering investigate the structure of modules (e.g. the separation of concerns, cohesion, and coupling). This work, however, focuses on the artifact itself as opposed to its use by its consumers. The API has also often been an incidental object of attention in studies of programmers (e.g., [22]). Thus, while useful information was gleaned, there was no coordinated theory building around API use and usability.

Recently, however, API usability has drawn more attention. In part, this is due to the increase in the number and size of APIs. Indeed, API documentation used to fit within small books whereas now it can fill multiple DVDs (e.g., the MSDN library). At the same time, the dominance of the use of APIs in everyday programming has also increased dramatically. It is now rare for anyone to write code that does not call APIs, and some tasks, such as user interfaces and graphics or the use of network resources and web-based services, cannot be performed without using APIs.

The Software Engineering Institute (as of 2008) defined the API as "technology that facilitates exchanging messages or data between two or more different software applications" [5]. Further, they categorize APIs as being remote-procedure calls (RPC), standard query language (SQL), file transfer, and message

delivery. While this restricted definition may be useful for some academic categorizations and taxonomies, it does not reflect common usage of the term.

A definition for API that more accurately maps to common usage is given in [12]: “any well-defined interface that defines the service that one component, module, or application provides to other software elements”. Thus, an API can be the interface for an object model your teammate wrote, a framework, a library, a toolkit, or a software development kit (SDK). Certainly, there are many significant distinctions that can be made between these types of APIs, but in essence the purpose and usage is the same: they each provide a programmatic user-interface to a module of code.

The earliest reported formal user study of an API of which we are aware was by McLellan and colleagues in 1998 [17]. The only direct follow-on work (as far as we know) was an MS thesis examining a specific API [4]. Other early work includes two books full of advice by API designers who worked on the Java APIs [2] and .NET framework [9]. One of these authors went on to spend significant time advocating further study of API usability, highlighting API design as a key factor in the success and failure of companies [3].

In the mid-2000s, Steven Clarke also began to conducting usability studies on APIs [7,8]. His work was not only reporting on significant APIs of widespread use (Microsoft), but also incorporating the use of the Cognitive Dimensions of Notation framework [14]. His work is also worthy of mention in part because it has directly inspired a plethora of work by Carnegie Mellon researchers led by Brad Myers and Jeff Stylos (e.g., [1,13,25,26,27,28]). They have quantified the impact of API design choices on users, sometimes showing time penalties of a factor of 3 to 10 [13,26].

Around the same time Clarke was pioneering API usability analysis at Microsoft, deSouza and colleagues reported on what we believe to be the first formal analysis of APIs from an organizational perspective [12], and Daughtry and colleagues examined aspects of API documentation [10] and polymorphism [15].

Following on the success of the earlier books described above, two new practitioner-focused books were released in the mid to late 2000s [21,29]. Each of these books seeks to describe API design as a craft that is based not only on structure, but also on usability.

With the growth that is taking place within the area of API usability, it is easy to forget that usability analysis of programming is not a new concept (e.g. [19,21]). With this history in mind, a special interest group session was held at CHI 2009 in Boston, MA [11].

Participation

Over 30 participants attended the CHI SIG. These participants represent the breadth of the software industry, representing very small firms, large software companies, academics, standards organizations, product-based business models, and service-based business models. Almost every attendant was actively engaged in both programming and usability analysis. There was a fairly even balance between those who were interested in the research side of API usability versus those who attended due to their frustrations using APIs.

A Taxonomy for APIs

Throughout the session, we heard from attendees who expressed an interest in the peculiarities of specific API domains. Examples include:

- Robotics: An API mapping to a controllable physical thing
- Mash-ups: An API that supports tentative loose integration
- Artificial Intelligence: An intelligent API
- Distributed Systems: A remote API
- Services: An API expressed in many languages and platforms
- Open Source: An API developed by a distributed team
- Embedded Systems: An API tightly coupled with hardware

Another issue raised was that there needs to be a more formalized distinction between the different levels of APIs. On the one hand, we often speak in terms of the larger framework or platform APIs such as the .NET framework or the Java JDK. However, there are other types of APIs ranging from internal APIs to platform APIs. At various points along this path of scale, different problems and opportunities arise with respect to API usability. For example, a larger platform API may be less changeable, while at the same time it is significantly easier to find documentation related to its usage. What are the critical points along this path of API scale? What constraints and opportunities exist that relate to API usability along this scale?

Usability Is Multifaceted

Participants reported problems dealing with various aspects of usability, including learnability and testability. One problem in the discussion that was surprising, given the audience, was the tendency to black-box usability as a single attainable quality. Indeed, if there is one thing we can learn from the CHI community is that usability is a multi-faceted set of trade-offs as opposed to a singular, objective, and attainable end goal. For example, there is a trade-off between the simplicity and the power of an API. Certainly, there are abstractions that allow an API to be simpler, but the fundamental trade-off still exists to a degree. Further, usability is subject to the culture in which it is embedded and the context of its use within that culture.

Role Models Needed

One area of concern to several attendees is the lack of role models in for API design. Simply put, a young engineer is exposed to many APIs, but none are presented to him as paragons of API design. These exemplary designs may exist, yet they are not identified, studied, and shared with novices and young professionals as case studies. As a design profession, it seems that collecting examples of APIs with good usability, and case studies of good API design practices would be worthwhile.

API Evolution

We heard from an engineer at a large software company who was engaged in the publication of a new API for a large commercial product. When it was decided that the API should be released to the world, the engineers appear to have simply opened up access to the existing internal API that they had been using for quite some time. The API in question is now well-regarded and very successful. This simple example raises issues with respect to the evolution of APIs over time. In what ways do different APIs evolve over time? What are the biggest threats with respect to

aspects of API usability given a particular development path? What constraints or opportunities arise with each path?

Which APIs Necessitate Usability?

We heard from a program manager about his first exposure to API usability. While working on a project, he was approached by the test team, who expressed fury over the product's public-facing exposed APIs. They complained that the APIs were difficult to understand and impossible to test, yet the developers would not address the issues because it was a programmer's interface as opposed to a user's interface. How do developers perceive the role of the public facing API within the context of the larger business process? How can we change that perception by making developers intrinsically motivated to produce usable APIs?

We also heard a similar story from another participant. In this case, the developers argued against changing the API because it was not a supported public API. Rather, it was an API that end-users had found and were using without approval (or significant objection) by the company that makes the software. This raises several issues with respect to the business of API use. What are the trade-offs between documentation, usability, exposure, and use? If an API is publicly accessible, does a company have an ethical or legal responsibility to make it as usable as possible?

Another issue is backwards compatibility. In the experience of many participants, public APIs are withheld from users far longer than is perhaps useful or required for the sole problem that supporting an API extremely expensive over the long-haul, so designers want to try to make it "perfect" before release. This seems to be an unsatisfactory resolution for both API designers and API users. There exist mechanisms for API evolution that afford the eventual retirement of capabilities (e.g., the Java deprecated keyword). What mechanisms might exist for helping to facilitate graceful API evolution at the beginning of the lifecycle as opposed to just at the end?

Interdisciplinary Involvement

In all, it was a fruitful and lively discussion that reinforces the argument that there is ample opportunity for future work that is of value and interest to both the HCI and SE communities.

As of May 2009, a new website has been started at www.apiusability.org. On this website, you can find a list of people conducting research related to API usability and a collection of links pointing to the material on API usability that we have found. The site is managed using Google Sites and collaboration in terms of the content is encouraged. Please visit the site and participate in a small but growing community of researchers interested in API design and usability from both the realm of SE and HCI. In particular, please contribute to and subscribe to the lists of publications and resources that deal with API usability.

Acknowledgements

The authors thank participants of the CHI 2009 SIG on API Usability. The authors affiliated with Carnegie Mellon University are supported by funding from various sources, including Microsoft, SAP, and the National Science Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors or the authors' respective institutions.

References

[1] Beaton, J., Jeong, S.Y., Xie, Y., Stylos, J., and Myers, B.A. "Usability Challenges for Enterprise Service-Oriented Architecture APIs". Proc. VL/HCC 2008, IEEE Press, pp. 193-196.

[2] Bloch, J., Effective Java: Programming Language Guide, Addison-Wesley, 2001.

[3] Bloch, J., "How to Write a Good API and Why it Matters", Keynote Address for LCSD workshop at OOPSLA, 2005. Introduction and slides available at http://lcsd05.cs.tamu.edu/#keynote.

[4] Brown, C.A. "Usability Analysis of the Channel Application Programming Interface". Unpublished Master's Thesis. Naval Postgraduate School. Monterey, CA, 2003.

[5] Carnegie Mellon Software Engineering Institute Software Technology Roadmap: Application Programming Interface. 2008. Web archive available at: http://www.sei.cmu.edu/str/str.pdf.

[6] Cherubini, M., Venolia, G., DeLine, R. and Ko. A. J. (2007). "Let's Go to the Whiteboard: How and Why Software Developers Draw Code", Proc. CHI, 557-566.

[7] Clarke, S., API Usability and the Cognitive Dimensions Framework. 2003. Available at http://blogs.msdn.com/stevencl/archive/2003/10/08/57040.aspx.

[8] Clarke, S., "Measuring API Usability". Dr. Dobbs Journal, May 2004, S6-S9.

[9] Cwalina, K., Abrams, B., Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET libraries, Addison-Wesley, 2005.

[10] Daughtry, J.M. "Enabling Use: The Effects of Scenarios and Claims on the Self-Efficacy of Using Application Program Interfaces". Unpublished master's thesis, The Pennsylvania State University, University Park, 2006.

[11] Daughtry, J.M., Farooq, U., Stylos, J., Myers, B. "API usability: CHI'2009 special interest group meeting", Proc. CHI 2009 (extended abstracts), ACM Press, 2771-2774.

[12] de Souza, C.R.B., Redmiles, D., Cheng, L., Millen, D., and Patterson, J. "Sometimes You Need to See Through Walls – A Field Study of Application Programming Interfaces", *Proc. CSCW*, 2004, 63-71.

[13] Ellis, B., Stylos, J., and Myers, B.A. "The Factory Pattern in API Design: A Usability Evaluation", Proc. ICSE 2007, ACM Press, pp. 302-312.

[14] Green, T. R. G. "Cognitive dimensions of notations. People and Computers" V.A. Sutcliffe and L. Macaulay, eds. Cambridge: Cambridge University Press. 1989.

[15] Kannampallil, T.G., and Daughtry, J.M. "Handling Objects: A Scenario Based Approach", Proc. SIGDOC 2006, ACM Press, 92-98.

[16] Norcio, A.F., "Indentation, Documentation, and Programmer Comprehension", Proc. CHI, 1982, 118-120.

[17] McLellan, S.G., Roesler, A.W., Tempest, J.T., and Spinuzzi, C.I., "Building More Usable APIs", IEEE Software, 15(3), 1998, p. 78-86.

[18] Myers, B.A, Bernett, M.M., Rosson, M.B, Ko, A.J., and Blackwell, A, "End User Software Engineering: CHI'2008 Special Interest Group Meeting, CHI 2008 Extended Abstracts, 2008, 2371-2374.

[19] Patel, K, Fogarty, J., Landay, J.A., and Harrison, B., "Investigating Statistical Machine Learning as a Tool for Software Development". Proc. CHI, 2008, Florence, Italy, pp. 667-676.

[20] Pemberton, S., "Programmers are Humans Too, 2" SIGCHI Bulletin, 29(3), 1997, p. 64.

[21] Pugh, T. "Interface Oriented Design". Pragmatic Bookshelf. 2006.

[22] Rosson, M.B., and Carroll, J.M., "The Reuse of Uses in Smalltalk Programming", ACM Transactions on Computer-Human Interaction, 3(3), pp. 219-253.

[23] Shneiderman, B., Software Psychology: Human Factors in Computer and Information Systems. 1980, Cambridge, MA: Winthrop Publishers.

[24] Soloway, E., Ehrlich, K., and Bonar, J., "Tapping into tacit programming knowledge", Proc. CHI, 1982, 52-57.

[25] Stylos, J. and Myers, B.A. "Mapping the Space of API Design Decisions", Proc. VL/HCC 2007, IEEE Press, 50-57.

[26] Stylos, J. and Clarke, S., "Usability Implications of Requiring Parameters in Objects' Constructors", Proc. ICSE 2007, ACM Press, pp. 529-539.

[27] Stylos, J., Clarke, S., and Myers, B.A. "Comparing API Design Choices with Usability Studies: A Case Study and Future Directions", Proc. PPIG 2006, pp. 131-139.

[28] Stylos J., Myers B., and Yang Z. "Jadeite: improving API documentation using usage information", Proc. CHI 2009 (extended abstracts), ACM Press, pp. 4429-4434.

[29] Tuloch, J. Practical API Design: Confessions of a Java Framework Architect. Apress. 2008.