Using a Distributed Heuristic Evaluation to Improve the Usability of Open Source Software

Alexander Faaborg
Mozilla
650 Castro Street Suite 300
Mountain View CA, 94041 USA
faaborg@mozilla.com

Daniel Schwartz
Oracle
500 Oracle Parkway, MS2op10
Redwood Shores, CA 94065 USA
daniel.schwartz@oracle.com

ABSTRACT

Building tools to enable a large scale distributed Heuristic Evaluation of usability issues can potentially reshape how open source communities view usability, and educate a new generation of user experience designers. We are exploring adding the ability to perform a distributed Heuristic Evaluation into the Bugzilla instance used to develop Firefox. Ideally this approach will build more of a culture around HCI principles, and will create a framework and vocabulary that will cross pollinate to other open source projects.

Author Keywords
Heuristic Evaluation, Open Source, FLOSS

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION

When contemplating how to increase the influence of user experience design in open source communities, a common approach is to attempt to "increase the involvement and visibility of UX professionals" [1]. However, this paper asks a different question: how can we convert current developers in open source projects to have a skill set equivalent to what academia or a corporation would consider to be formally trained user experience professional? The approach we propose consists of embedding HCI concepts and practices directly into the tools that control an open source community's work flow.

Beyond controlling an open source community's process and work flow, tools also indirectly shape the community's values and ideals. This is important, because if social currency in the community is inherently linked to one's ability to make the software better, the concept of better must be expanded to encompasses "easier to use."

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CHI 2010, April 9–16, 2010, Atlanta Georga, USA.
Copyright 2010 ACM 978-1-60558-246-7/09/04...$5.00.

QUANTITATIVELY MEASURING USABILITY

Measurements like the time it takes an application to load, the amount of memory used, or load on the cpu are all trivial to calculate, and wonderfully quantitative. One of the reasons open source communities tend to discount usability (both in practice and in artifacts like the severity descriptions in Bugzilla), is an inaccurate view that usability is an amorphous, and subjective thing that simply can't be scientifically quantified and measured. However, measuring an application's usability is an area where previous HCI research can make a strong and very significant contribution to open source development.

The usability inspection technique of Heuristic Evaluation, which was introduced by Jakob Nielsen [2,3,4] has emerged as one of the most common ways for professional user experience designers to evaluate the usability of a software application. Heuristic Evaluations are extremely useful because they formally quantify the usability of a software application against a set of well defined and irrefutable principles. Usability violations can be quantified individually: either an interface *supports undo*, or it does not, either an interface is *internally consistent*, or it is not, etc. Usability violations can also be quantified in aggregate: *the software application currently has 731 known usability issues*. Additionally, by building the tracking system on a set of agreed upon principles, much of the debate on the level of "there is no right or wrong with UI / every user is entitled to their personal opinion / all that matters is the ability to customize" which is currently found in open source software development communities may be significantly reduced. Usability heuristics will help ground these debates, just as currently no one in an open source community argues in favor of *data loss*, or in favor of *crashing*.

INJECTING HCI PRINCIPLES INTO BUGZILLA

Adapting an open source community's bug tracker to capture usability issues defined by a set of specific heuristics can reshape the way developers think about usability. Just as open source development communities currently have a shared vocabulary to describe good and bad with concepts such as *performance*, *data loss*, and *crashing*, usability heuristics can introduce additional concepts, like *consistency*, *jargon*, and *feedback*. All of these concepts, covering both the underlying implementation as well as the user interface, can now have an equal potential to impact the software application at any level of severity, from trivial to critical.

Modifying a bug tracking system to track a Heuristic Evaluation of software is reasonably straightforward. Each issue needs to be able to be associated with the specific usability heuristic being violated (for example: "using the term POSTDATA in a dialog is technical *jargon*"). We plan to utilize Bugzilla's keyword functionality, similar to how current bugs can be flagged as violating implementation level heuristics, like *data loss.* Since the evaluations will be performed by contributors who likely will not have any additional interface design training, it is important that each heuristic is very clearly defined with specific examples and detailed explanations. Additionally, allowing contributors to view all of the bugs in the software marked as the same type of issue, both current and resolved, serves as an effective way for them to further learn about the heuristic.

We are now working to embedded to functionality needed for a distributed Heuristic Evaluation into the Bugzilla instance used to develop Firefox and Thunderbird. These specific modifications may spread to a variety of other open source projects as Bugzilla is currently used by communities including the Linux Kernel, Gnome, KDE, Apache, Open Office and Eclipse [5]. Ideally embedding HCI principles into development tools will also embed the ideals into the community. Similar to other forms of bugs, there will be a social incentive for contributors to locate and classify violations, and there will be a social incentive for other contributors to resolve them. As open source contributors travel between different communities and projects, the usability heuristics will also see a similar cross pollination between open source communities. A shared vocabulary will emerge across open source projects, allowing for clearer communication and debate.

Side Effects of Distributed Heuristic Evaluation

Today the process of Heuristic Evaluation is normally completed in corporations and academia by a small number of designers, who are extremely well practiced at identifying usability issues. However, it is worth noting two important aspects of the Heuristic Evaluation method from when it was first introduced:

Education - First, the method of Heuristic Evaluation has its roots not in the functional purpose of evaluating usability, but rather in the even more basic purpose of teaching usability. We see this in Nielsen's 1989 SIGCHI bulletin: *Teaching User Interface Design Based on Usability Engineering* [2] that Heuristic Evaluation was introduced as part of the curriculum for a masters degree in Computer Science. This is still true today: the road to becoming a good user experience designer begins with mastering the identification of well defined heuristics.

Power in Numbers - The second important aspect of Heuristic Evaluations is that it was quickly found that the number of evaluators played a major role in how successful it was. Nielsen wrote in 1990 that "evaluators were mostly quite bad at doing such heuristic evaluations... they only found between 20 and 51% of the usability problems in the interfaces they evaluated. On the other hand, we could aggregate the evaluations several evaluators to a single evaluation and such aggregates do rather well" [3]. For large open source projects, Bugzilla instances often have thousands to hundreds of thousands of users.

CONCLUSION

In open source software development, the educational and distributed aspects of a Heuristic Evaluation are critically important. While the majority of open source projects currently lack user interface designers capable of performing a perfect Heuristic Evaluation in isolation, that's irrelevant. The collaborative nature of open source projects allows for a group of contributors to effectively compete with a formerly trained user experience professional by aggregating their abilities. And similar to all of the other ways in which people contribute to open source projects, there is a mutually advantageous feedback loop: the more effort a contributor puts into improving the software, the more they are able to increase their own skill set. Hours spent performing heuristic evaluations and brainstorming ways to address usability issues will allow a new generation of user experience designers to emerge in open source communities, just as rock star software develops are currently forged there.

REFERENCES

1. Schwartz, D. and Gunn, A. 2009. Integrating user experience into free/libre open source software: CHI 2009 special interest group. CHI EA '09. ACM, New York, NY, 2739-2742.
2. Nielsen, J. and Molich, R. 1989. Teaching user interface design based on usability engineering. *SIGCHI Bull.* 21, 1 (Aug. 1989), 45-48.
3. Nielsen, J. and Molich, R. 1990. Heuristic evaluation of user interfaces. CHI '90. ACM, New York, NY, 249-256.
4. Nielsen, J. 1994. Enhancing the explanatory power of usability heuristics. CHI '94. ACM, New York, NY, 152-158.
5. Bugzilla Installation List, http://www.bugzilla.org/installation-list