# Open Source Interface Politics: Identity, Acceptance, Trust, and Lobbying

**Roshanak Zilouchian Moghaddam**
University of Illinois
Urbana, IL
rzilouc2@illinois.edu

**Kora Bongen**
University of Illinois
Urbana, IL
kbongen2@illinois.edu

**Michael Twidale**
University of Illinois
Urbana, IL
twidale@illinois.edu

## ABSTRACT

A study of the Drupal open source project shows the rather problematic status of usability designers with respect to the larger developer community. Issues of power, trust, and identity arise and affect the way that usability recommendations are acted on or ignored. A political view of these aspects can help in interpreting the situation. We found that making a straightforward case for a particular interface design can be insufficient to convince developers. Instead various additional lobbying strategies may be employed to build up a quorum of support for the design.

**Author Keywords**
Open source, usability, politics, lobbying.

**ACM Classification Keywords**
H5.2. User Interfaces: User-centered design

## INTRODUCTION

Various open source projects have made efforts to improve the usability of their application by recruiting usability experts. However this does not magically solve the problem. Similar to experiences in commercial software development, a usability expert can face challenges of persuading the organization as a whole and software developers in particular of the value of adopting good usability practices. Despite the implications of the religious and legal metaphors of "usability evangelist" or "user advocate", this is not simply a matter of a charismatic individual making a compelling rational or empathic argument and the audience being persuaded. Additionally there are various inherently political aspects of assembling alliances and appealing to different sub-groups' mutual self-interest. The case is typically not made in a moment, but requires an ongoing series of arguments, and indeed lobbying of others to help continually make the case. This can be made more difficult when usability experts are not fully accepted as equal members of the larger community. In this paper we consider these issues in the context of the Drupal project.

## INTERFACE DESIGN IN DRUPAL

Drupal [4] is a content management platform which started in 2000 and released as an open source project in 2001. Anyone can contribute to the Drupal core code by submitting a patch. A contributor opens an issue for a patch in the issue queue (a database of bugs and feature requests), or she can submit a patch for an open issue. The issue queue is monitored by the community. Patches submitted to the issue queue are peer reviewed and then either Drupal's founder or one of the core committers, who have write access to the Drupal code repository, decides on including the patch in the next version. Drupal is a developer dominant community, but in the past four years several user interface (UI) designers have joined including two employed to work on Drupal.

As part of a larger study of interface design in a number of different open source projects we interviewed seven designers working on Drupal. One question in particular triggered the issues we report here: "What are the main challenges in the process of designing an open source UI?" Initial findings from the interviews were supplemented by studying usability related issues in the Drupal issue queue, posts in the usability group forum, and certain Drupal designers' blogs. We quote from these sources below.

Most informants reported substantial barriers in getting the ideas of usability accepted within Drupal. Many challenges were similar to those observed in introducing usability into commercial software development settings [2, 3]. Usability can be misunderstood as pointless, an expensive overhead, something that can be done at the last minute, all about making the interface look nice, hopelessly subjective, or a distraction from what is considered the main objective of delivering working code with powerful functionalities.

## MAKING THE CASE FOR USABILITY

In the interview most designers mentioned as a main challenge the need to convince developers that a proposed design is worth implementing. A Drupal designer who has been in the community for 3 years commented: "Everybody has an opinion on designs [...] It's hard to keep discussions on track, because by nature design discussions are pretty broad in scope. [...] It's hard to convince developers."

Because designers do not create code, they have to persuade a developer to implement a design: "We had to get the people, the developers from the community, on board with our design approach because they were the ones who were going to build it and if they aren't motivated to build it, it wouldn't get built."

The problem of convincing developers exists partially because developers lack familiarity with designers' work: "The vast majority of people within the community don't have much of a history or background or kind of real empathy for the work that designers do and the way that they design - and the challenge is getting them to see why what you are doing is important."

## DESIGNERS VERSUS DEVELOPERS

Unlike many open source projects which may have no usability experts or just one, Drupal enjoys several, and support for greater usability comes from the highest source – the project founder. Nevertheless, problems of identity remain, and something of an us-versus-them attitude made visible in the use in the project of two distinct terms: developers (those who develop code) and designers (those who argue for usability and specify and draw interface elements but typically do not code them).

Drupal is a developer dominant community. There are 2416 people who have contributed to Drupal modules whilst 184 people use the "usability" tag. Given the large variance in levels of contribution, and the fact that some developers also use the usability tag, these numbers only give a very rough sense of relative proportions. As another indication, one respondent claimed there were 4-5 active designers compared to 300 developers.

Developers used to implement different functionalities for Drupal without seriously considering their associated usability issues. Since developers built the whole system and contributed code to Drupal, they feel ownership over Drupal. Although some of them accept designers as part of the community, a lot of them do not want designers to change or throw away the code they have implemented. A similar situation was witnessed by Mirel in a commercial setting, where designers were brought in from the outside and clashed with an existing developer culture [6]. There is also a clash with the ethos of open source development which encourages adding your own code but has problems with advocating the deletion of others' code or functionality on the grounds of usability through simplicity.

There are numerous explicit developer-designer conflicts in the issue queue. In an issue about removing the breadcrumb navigation and not displaying a menu, a developer wrote a comment in which he said that he disagreed with the proposed design change. He then adds "D7UX [the Drupal version 7 usability experience team] are not the Drupal community. They are missing the point on a lot of usability issues, which are well proven. I believe that we must improve on their design and not be stuck with their shortsightedness."

Many designers see cultural differences between developers and designers as a challenge. One designer, involved in Drupal for seven years said: "They are not much recognizing the importance of design and there is not enough communication." The concern with communication is widespread amongst designers. One subpart of that concern is that the issue queue (the main communication channel in Drupal) is not ideal for discussing issues about interface design and the user experience.

One issue reported recurs in many open source projects [7]: "One of the biggest challenges for the Drupal project is that the people who we were designing for, were not necessary the same as the people who were most active within the community. … people in the community had been designing for themselves for a long time and that had resulted in an interface that had become pretty much unusable to anybody else."

Designers and developers can seem to represent different constituencies - designers often speaking on behalf of novice or less technically sophisticated users (or intended potential users) and developers speaking on behalf of power users (very much like themselves) When this clash leads to contradictory design implications we see discussions such as this: "For newcomers I assume X's proposal suits them well, but for advanced users, like Drupal developers (which is 80% of the users who visit the module page), this surely isn't preferable." Resolution in such circumstances may revolve around relative weighting of the importance of the issue. In this case the degree to which the two constituencies would use that interface element is used in the argument.

## TALK IS SILVER, CODE IS GOLD

Within a discussion about the use of high and low fidelity prototypes to communicate with developers, one designer noted: "Historically there is a saying in Drupal community that *talk is silver, code is gold* meaning, you know, show me your code first and then we start talking [...] whatever improvements you try to make the best way to communicate those ideas is to provide actual code."

This slogan is widely quoted in the Drupal community. As such it emphasizes the problematic status of designers who just talk and do not code. They are using a different, devalued currency and so can struggle to make their case heard amongst competing arguments for code ideas from actual coders.

A Drupal developer wrote in his blog: "The developer writes the code and ultimately gets a piece of functionality, whatever it is, to work. In fact, the services of a designer are never required to make this code work. The fact that the services of a designer is a really good idea doesn't really come into this... The converse, however, is not true. If a designer desires a particular piece of functionality, the services of a developer are required... Design isn't as easy to abstract and make into reusable components the way code can. Designers, in general, have less to contribute not because they do less, but because the volume of work that designers do isn't reusable. There's no point to contributing non-reusable work. That isn't what open source does."

A consequence of not coding is a lack of power. Designers struggle with a lack of ownership over their design: "... You have no ownership over your design [...] you can make something and then go like "well this is how it should be implemented" and then the community just [unclear] with it and change all kinds of stuff [...] and your table becomes like six pages long but that's not what I intended."

## A Do-ocracy

Much is made of the meritocracy of open source projects. But again, a distinction is made between those who talk and those who do (where 'doing' seems to only mean coding):

"There are a couple of propositions that are regularly thrown at me as a designer working on Drupal.org and d7ux. The first is: Ah, but that won't work with contributes modules. I call this the contrib grenade. It's normally thrown in when someone doesn't agree with your design direction and they're using the power of contribution the very life blood of open source as ammunition for their argument. The second is: It's a do-ocracy. Either contribute, or get out of the way. And, in there lies the problem."

## ACCEPTANCE AND TRUST

There seem to be numerous issues about designers being accepted by developers, and their suggestions being trusted. One of the designers employed to work on Drupal blogged about this: "Drupal 7 is the fruit of developer labour. And lots of it. For a designer to even enter the fray requires trust on behalf of the developer community. And buckets of the stuff. As one developer put it: 'You've come into our front room, and, while we were making a cup of tea, you moved all the funiture around. Not only that, but you redecorated, changed the carpet, and removed all of our belongings.'"

Typically in open source projects, acceptance and trust is built up over time, usually accompanied by an apprenticing mode. Ducheneaut identified six steps to becoming a developer (in Python) [5]. Along the way, the developer must prove herself both technically and socially. Technical skill is not sufficient for advancement. The developer must navigate through the community by starting humble, gathering allies, and offering gifts of code. Code is often considered a form of currency in open source software. People give code as a way to gain power and recognition[1]. Before they do that they have to uncover the hidden community social structure and where in the community they might fit as well as how things are done. There can be very little help in how to do this. Clearly if you do not code you lack a crucial currency.

There is evidence that the Drupal community is not particularly friendly to newcomers. A designer noted: "We are … [trying to] open those things up and make them more public and try to encourage some more people in, but honestly you know I think the chances of getting a lot of people to engage in that issue queue is fairly [unclear] because it's a hostile environment." There is something rather ironic (but not that unusual [5]) about an open source community that is not particularly open to new members. Gaining acceptance requires passing the 'test' of understanding the community and its history as well as demonstrating the ability to make valued contributions. Even developers can struggle to be accepted as can be seen by the relatively few numbers who make it to the upper levels of the hierarchy. At times it even seems as if we have the situation of 'Open Source – Closed Community'.

Many of the problems in Drupal may be due to it going through a transition of an influx of usability people into a developer-centric community. The paid usability experts may be in an especially awkward situation, being expected to justify their salaries by dramatic interventions but in the process being seen to be disruptive of the expectation of a gradualist earning of trust and acceptance [5].

## LOBBYING

Given all these challenges, designers cannot simply make a case for a usability improvement and expect that the argument will stand alone. As noted there are issues of power and authority. In line with this political perspective, it is not too surprising that we see activities analogous to lobbying. That is, recruiting people to support your idea, including drumming up support to increase the number of supporters attending a meeting. Lobbying activities take place in all the forums used by Drupal designers to discuss design issues: IRC channels, the issue queue, and the usability group forum. The following examples illustrate various kinds of lobbying.

When we asked one designer how they overcome the challenge of persuading a developer to develop the solution he replied: "We beg, "please" and we make a lot of noise [...] We post a lot of comments and we write to people and we start new discussions and we say very loudly that this is a big problem. We organize discussions like Skype meetings, Net Meetings outside of Drupal."

A blog post about being an open source designer notes the challenge of a design that has been in the works for several months, and is starting to change direction. Referring to another designer he notes: "I can't be sure, but I'm hoping he's rallying the troops to weigh in so that the design stays on track."

Similarly another designer noted "Very often in the issue where these decisions are ultimately made there'll be 40 developers and maybe two designers [...] so that's a tough battle [...]" Clearly if usability discussions were purely rational, it should not matter that the usability advocates are outnumbered 20 to one. And yet it is easy to see that such socio-political issues do matter.

Various strategies are used to add numerical weight to a case. Designers try to participate in different usability related discussions and vote in favor of a design change to show the community that the proposed change is accepted by more than one person: "You are going to IRC and [ask other designers] 'Please come in and comment on my issue.' What you see in the issue queue is generally a fraction of the discussion that is actually going on."

Issue postings can contain references to IRC discussions where prior persuasion has occurred, e.g: "X's proposal is basically the summary of an IRC discussion I had with him about this. So I'm in favor of this patch as well." Similarly from another issue: "I'm not in the IRC chat summary above but was part of it, meaning, I'm in favor of this happening." In that same discussion, another designer reported "Just want to register my support for this."

Even when the case has been made to a developer to code the design, further work can be needed to include the patch. Code committers will not include the patch until they are convinced that the code is good: "It's reputation: sometimes I have to provide links to other research or blog post where [...] web professionals point out solutions to stuff or research. Sometimes I have to do that, sometimes I say "yes I think it's a good idea because this and this and this" and sometimes it [convincing the code committer] needs one of us [designers], sometimes three of us to do that. But we have building enough trust now that our thumbs up are enough." In this case we see the marshalling of four different resources to make the case: earned personal reputation; external research; blog posts by professionals; and additional Drupal designers weighing in.

Not all lobbying is or will be successful. Decisions have to be made about using up effort and political capital. In an issue about adding default values to a field, a designer suggests it should be an option and the creator of the issue agrees. But this will have an effect on another module. So, the developer asks the designer to convince contributors in that other issue that the change is a good idea. But the designer replies "I will just give up. Battling a security decision is rather useless, especially when you will also have to battle an opt-in discussion."

## CONCLUSION

This preliminary study reveals various inherent political processes in one open source project attempting to incorporate greater usability. It is not surprising that a collaborative activity dealing with scarce resources (people's time and willingness to work on a large number of competing development tasks) should have a political dimension. Issues of trust and acceptance arise that can be problematic when there is a need to accelerate a typically slow trust-building process. The various kinds of lobbying show how a political analysis can inform a deeper understanding of the design process and what would be needed to support it. We do not know how much our findings generalize to other open source projects and how much are due to Drupal being in a process of transition to embracing usability. It can seem to be inherent to open source that designers have to cajole developers to implement their designs. But do similar interactions occur in commercial projects? Also, is there any reason why so few designers actually create basic interface code such as screen layouts? Would such a relatively basic demonstration of coding skill help designers earn coding gold to influence future discussions?

It may be that open source projects need to support more explicit advocacy for the importance of usability as has happened in various commercial settings, while acknowledging that this is unlikely to be a one-shot solution. There are indications of a mismatch of expectations of what designers should 'produce' and why it matters. Finally, the existing tools, optimized for efficiently managing bugs and feature requests seem to be poor at supporting usability discussions and the creation of consensus around interface designs.

## REFERENCES

1. Bergquist, M. and Ljungberg, J. (2001)The power of gifts: organizing social relationships in open source communities. Information Systems, 11(4) 305-320.
2. Bloomer, S. and Croft, R. (1997). Pitching usability to your organization. interactions 4(6) 18-26.
3. Boivie, I., Gulliksen, J., and Goransson, B., (2006) The lonesome cowboy: A study of the usability designer role in systems development, Interacting with Computers, 18(4), 601-634.
4. Drupal, http://drupal.org
5. Ducheneaut, N. (2005) Socialization in an open source software community: A socio-technical analysis. Computer Supported Cooperative Work, 14(4) 323-368,
6. Mirel, B. (2000) Product, process, and profit: the politics of usability in a software venture. Journal of Computer Documentation. 24(4) 185-203.
7. Nichols, D.M. and Twidale, M.B. (2006). Usability Processes in Open Source Projects. Software Process - Improvement and Practice Journal: Special Issue on Free/Open Source Software Processes. 11(2) 149 - 162