About Us, With Us: The Fluid Project's Inclusive Design Tools

Colin Clark[1], Dana Ayotte[1], and Antranig Basman[2]

[1] OCAD University, Toronto, Canada,
cclark@ocad.ca
[2] Raising the Floor - International

Abstract. Since 2007, the Fluid Project has been developing an integrated set of inclusive design methods and software tools to support personalization, authoring, and software creation by users within the context of a participatory, open source community. In this paper, we position the Fluid Project's inclusive design practice within the context of interaction, participatory, and universal design methods, We examine and contrast these approaches from the perspective of supporting user creativity throughout the process of designing and using software. The Fluid Project is an open source community of designers, developers, testers, users, and other diverse contributors who might not otherwise fit into the highly technical and exclusive culture of conventional open source software communities.

Keywords: Accessibility, Inclusive design, Fluid Project, development tools, assistive technology, design methods, user creativity

1 Design Methods

1.1 Interaction Design

Industry-driven interaction design methods such as those of Cooper [3], Beyer et. al [1], and IDEO [8] primarily look inward; they are created by professionals and intended for an audience of like-minded designers and managers. These methods aim to provide prescriptive, generalized, and reproducible techniques for managing teams who design commercial software products or offer design consulting services. The emphasis is on "modelling" users, their goals, and their work or organizational processes (Beyer et. al [1]).

Despite an increased focus on user-centered design, which often invites users into the fold of research in a more active way (as with the human-centric design methods of [8]), such industrial modelling methods maintain the user in a passive role as "consumer" or "customer", often advocating for a rigid design focus on typical or mainstream requirements while explicitly de-prioritizing the "edge cases" of outlying, marginal needs While this approach works to simplify product requirements and focus designers on the most popular features, it also risks excluding the crucial features and customizations that enable people with disabilities to use a software product, and which ultimately contribute to greater innovation and the overall usability of a system [20].

Moreover, interaction design advocates often argue that their subjects (i.e. the individuals who use their software on a daily basis) are unable to be articulate or self-aware about their own technology needs, and that only industry can provide design innovation, not users themselves [19]. User input gathered through human-centered design methods provide a means for end-users to inspire the creative process of the "real" designers; the result is that there are few opportunities for individuals to actively contribute to the design process and work alongside professional designers, except as consumers or research subjects.

1.2 Universal Design

Universal design, with an explicit emphasis on meeting the needs of all individuals, including those with disabilities, substantially expands a designer's creative remit and responsibilities. However, the challenge of universal design is in its emphasis on a single product or design that aims to fit the needs of all users without adaptation or personalization. Ron Mace describes universal design as "the design of products and environments to be usable by all people, to the greatest extent possible, *without the need for adaptation or specialized design*" [our italics] [15]. As the complexity and diversity of today's software grows, we argue that it is no longer practical for designers to plan for every user and every feature within a single piece of software, nor to be able to fully understand and obtain expertise in the infinite variety of creative, serendipitous, and unexpected uses that software can be subjected to. Instead, the design process needs to be supported by technologies that provide users with a means to materially change, personalize, specialize, and extend their software environments.

1.3 Participatory Design

Participatory design, in contrast to many interaction design methods, offers the potential for users to more actively engage in the design process. This often takes the form of workshops and scenario-building exercises where users are invited to explore design strategies alongside professional designers [22]. While participatory and "co-design" techniques play a foundational role in the Fluid Project's inclusive design methods, particularly the concept of experienced designers working in harmony with users and other non-designers, we argue that workshops and other "before-the-fact" design methods alone are insufficient for three primary reasons:

1. Participatory design methods don't explicitly provide a means for ongoing community "stewardship" or "curation" of the software product after the initial participatory methods have been completed
2. Additional technological tools are needed in order support ongoing or user-continued design, including the ability for individuals to customize, adapt, and modify a "finished" software product

3. Most user-centered or participatory methods aren't inclusive in that they don't fully support the participation of users "at the margins" or with a wide range of needs

2 Fluid's Design and Technological Tools

Taking up the participatory design position that "all people have something to offer to the design process and that they can be both articulate and creative when given appropriate tools with which to express themselves" [18], the Fluid Project has been developing design and technological tools to support user creativity. We aim to extend the design process into the designed artifact itself — to give users the ability to continue the design process themselves, after the specialized design effort has been finished and the product has shipped. This approach to "adaptation as design" extends from creating tools that allow users to configure their own interfaces, to technologies that support remixing, repurposing and sharing.

Designing inclusively, we argue, requires more than just design processes, but also new technologies. This is the motivation for technologies such as Fluid Infusion[17], a software development framework that enables applications to be reconfigured in context and preference-sensitive ways, and the GPII Nexus[12], which provides a means to integrate diverse software components together in a way that can be supported by graphical authoring and programming tools accessible to non-developers.

2.1 Design Tools

Fluid's software and design tools are rooted in open community practices that emphasize the role of users — especially users with disabilities — as co-designers, "gardeners", and ongoing caretakers of the project's outcomes. Fluid's approach emphasizes the importance of non-prescriptive design methods and self-organizing collaborative teams who freely draw from a toolbox of design approaches (such as those documented in Fluid's Design Handbook [14] and the Inclusive Learning Design Handbook [13]) based on the design context and the needs of participants and project stakeholders. Some of Fluid's community-driven design methods include:

1. **UX Walkthroughs**, a hybrid technique based on heuristic evaluation and cognitive walkthroughs, which emphasizes paired or collaborative evaluation of user interfaces by designers and non-designers alike, and which serves to bring a diversity of perspectives to bear on the design process
2. **User States and Contexts**, which serve to "de-centre" and "multiply" personas, reducing the risk of stereotyping with personas by emphasizing the dynamic nature of a user and their needs across different contexts of use. This tool offers a way to represent and "query" or visualize a diversity of user needs and perspectives individually or in aggregate.

3. **Community design crits**, which bring together designers, developers, and users to discuss and critique design artifacts — including ideas, scenarios, mockups, and in-development software
4. **Open, transparent sharing** of design artifacts and discussion on mailing lists and other community forums based on lazy consensus governance principles [2], which support community-based decision-making processes.

These design and development techniques have been used with some success on a number of software design projects, and have been extended to other communities and open source projects such as the Global Public Inclusive Infrastructure [21]. However, there are significant challenges to designing within the context of open communities, many of which we are still exploring. In particular, meritocratic governance runs the risk of being exclusive and dominated by contributors in privileged positions (socially, culturally, technologically, and economically) if a system isn't in place to ensure that diverse contributions, especially those by non-coders, are recognized and promoted (Emke [7]). Access to open source collaborative forums, which are sometimes synchronous, require high bandwidth, or which privilege text-based communication over verbal or visual means, can limit diverse contributions. We continue to experiment with, evaluate, and test new social and technical methods for supporting ongoing engagement by individuals with disabilities, non-technical contributors, and those who might otherwise be excluded from conventional open source culture.

2.2 Technological Tools

Technological tools to support such open and interactive design processes are thin on the ground. This is not just because of the rarity of the processes they would be designed to support — we argue that the underpinning is for them is missing even at the technological level. It's worth surveying a parallel story of technologies designed to support the technical elites — software developers — in order to get a flavour of what support we imagine there could be for the design process.

Distributed Version Control for Software Developers Distributed version control systems (DVCS) gradually took over at the expense of traditional centralised version control systems during the 2000s. These emphasise a more democratic, 'peer to peer' model for managing community assets, rather than the previous authoritarian model whereby the definitive version of a community asset was stored in a nominated central repository. The system currently with the greatest mindshare is git[10], designed by Linux architect Linus Torvalds in response to problems he faced managing a very large source code base of millions of lines of code shared amongst thousands of contributors. Especially as hosted by the hugely popular infrastructure site GitHub, this creates hugely important new affordances for software developers, above those directly implied by adopting an open source contribution model.

Amongst the most interesting for our purposes is GitHub's *review model*. This allows a suggested contribution to a project to enter a rich interactive lifecycle centred around a git artefact known as a *pull request* — a request by the contributor that their work is accepted (pulled) into the community's shared project. GitHub's pull request review interface provides the following facilities to contributors:

1. **Difference focused** The interface highlights the differences between the project's state before and after the contribution is accepted. It's possible to navigate to complete views of project artefacts before and after the contribution.
2. **A Content-focused discussion** Comments can be attached to particular parts of the contribution. These can start off a dialogue between the contributor and other project members which remains attached to the particular content it is relevant to.
3. **An archived record** Even after the contribution is accepted into the project (or, perhaps rejected) the pull request interface remains permanently available in the archive, making it easy to revisit and understand previous discussions

It's worth noting that this pull request workflow is particularly a facility of GitHub rather than simply git itself — although it is crucially enabled by the core technology of git. Unlike git itself, GitHub is not an open source project and its free-tier facilities are provided to the open source community as part of a wider commercial offering by the GitHub corporation. However, alternatives to Git and free alternatives to GitHub exist, such as Gitorius on git, or the darcs hub[5] on darcs (although this currently has no pull request UI).

Nothing for Designers Nothing similar to GitHub's pull request UI and workflow exists to facilitate contributions to open source culture from non-developers. Our design teams currently exchange designs as binary files (Adobe Illustrator files, Adobe Photoshop files and PDFs) deposited in dropbox[6], as attachments to wiki pages, or sometimes using the non-distributed SVN version management system. Comments and discussion must be separated from the design artefacts and occur in side-channels such as emails, IRC chats and Skype messages. Not all of this discussion can be easily archived, and when it is, it is still very easy for it to become separated from the design artefacts in question. None of the design tools most commonly in use have accessibility features to facilitate contributions from individuals with disabilities. This huge discrepancy between the quality of tools provided to different kinds of contributors makes a further nonsense of the commonly bandied ideology of a "meritocracy" in open source communities.

Fluid's Dream of an Inclusive Tool Chain We dream that the same community affordances can be extended to all kinds of contributors. The barriers to this are formidable, both technical, economic, and social. Providing, for example, an accessible equivalent to Adobe's Illustrator or similar visual design tools is a prohibitive task. Even where open source alternatives exist to popular corporate visual design tools, they are no more likely to provide accessibility features, and the space of collaborative vector-based design tools integrated with robust, decentralised version management is virtually empty even in the corporate space. It is clear that such tools can never be developed unless there can be a fundamental change to the economics of software development. Such a collaborative, accessible design tool could never stand alone as a single product, but could only be viable as the pinnacle to a broad pyramid of inclusive tools meeting a spectrum of similar needs, and meeting the broader needs of inclusion and collaboration at each infrastructural level.

3 Building an Inclusive Tool Chain

Forming the basis of such an inclusive tool chain is the core aim of Fluid's core framework, Infusion. The needs of inclusion and collaboration both induce similar kinds of requirements on software and content creation processes.

3.1 Working with Design Landmarks

A crucial way of organising such requirements is in terms of **landmarks** in a design — named architectural points which can be used to focus discussion as well as used to target further design. We take our inspiration from the kinds of landmarks visible in HTML documents which can be successfully referenced by means of a vocabulary of *CSS selectors* [4]. Landmarks can, for example, take the form of *tag names*, *CSS class names* or other attributes of a document node. By means of targetting these landmarks with CSS selectors, designers can meet both the needs of styling (using CSS rules), as well as further authoring (by using document-oriented manipulation tools such as the jQuery library). Landmarks are also a crucial requirement for a collaborative design process, in order to attach comments and document-directed discussion on the design. Such CSS rules have been an early, basic success for inclusive design, allowing designers and developers to share access to a design space in a harmonious way.

Infusion aims to bring the affordance of these selectors and rules to the software creation process, by organising software in *cellular units* named *components* which are analogous to the DOM nodes underlying an HTML document, and targetting them by means of **IoCSS rules**, analagous to the CSS rules used for styling and editing HTML documents. This casts the world of software development more in terms of document authoring than source code editing, and its final artifacts taking the form of structured trees with landmarks rather than binary opaque blobs. By creating an infrastructure which is suitable for authoring such trees, and for casting designs both visual and architectural in terms of them, we will support the creation of inclusive, collaborative workflows for designers of all kinds.

3.2 Requirements Beyond Frameworks

However, not all such needs can be met simply through the means of a development framework — other requirements need to be met through

- Building up hosting infrastructure for applications and data, and the tools and infrastructure needed to manage it
- Building up community structures and workflows for welcoming, supporting and reviewing contributions
- Building up and curating shared understanding of productive ways of casting and solving design and implementation problems, organised, for example, in "handbooks" of design guidelines, "bestiaries" of problems and their solutions or other approaches like those listed in section 2.1

4 Conclusion

We will build marvellous things that will support inclusive, collaborative design processes that will allow users, designers, developers and other contributors to take control of their artefacts and the tools they use to create them.

References

1. Beyer, Hugh and Karen Holtzblatt. *Contextual Design: Defining Customer-Centered Systems.* San Francisco: Morgan Kaufmann Publishers, 1998.
2. Capra, Eugenio and Anthony I. Wasserman. *A Framework for Evaluating Managerial Styles in Open Source Projects.* Open Source Development, Communities and Quality. Ed. Barbara Russo, et al. IFIP International Federation for Information Processing, Volume 27. Boston: Springer, 2008. pp. 1-14.
3. Cooper, Alan and Reimann, Robert and Cronin, Dave: *About Face 3: The Essentials of Interaction Design.* Indianapolis: Wiley, 2007.
4. Mozilla Developer Network - Getting Started with CSS Selectors `https://developer.mozilla.org/en/docs/Web/Guide/CSS/Getting_started/Selectors`
5. darcs hub - simple version control and collaboration `http://hub.darcs.net/`
6. Dropbox works the way you do `https://en.wikipedia.org/wiki/Dropbox_(service)`
7. Emke, Coraline Ada. *The Dehumanizing Myth of the Meritocracy.* Model View Culture (21), 2015. `https://modelviewculture.com/pieces/the-dehumanizing-myth-of-the-meritocracy`
8. "Methods." *Design Kit.* `http://www.designkit.org/methods`
9. GNU Image Manipulation Prograsm `https://www.gimp.org/`
10. The `git` project `--distributed-is-the-new-centralized` `https://git-scm.com/`
11. The `GitHub` project `https://en.wikipedia.org/wiki/GitHub`
12. The GPII Nexus API `https://wiki.gpii.net/w/Nexus_API`
13. The FLOE Project, *The Inclusive Learning Design Handbook* `http://handbook.floeproject.org/`
14. The Fluid Design Handbook `https://wiki.fluidproject.org/display/fluid/Design+Handbook`

15. Mace, R. et al *The Principles Of Universal Design* `https://www.ncsu.edu/ncsu/design/cud/about_ud/udprinciplestext.htm`
16. Clark, Colin, et. al.: *Preferences Framework Overview* `http://wiki.gpii.net/index.php/Preferences_Framework_Overview`
17. Fluid Project: *Fluid Infusion combines JavaScript, CSS, HTML and user-centered design*: `http://fluidproject.org/products/infusion/`
18. Sanders, Elizabeth B.N. *From User-Centered to Participatory Design Approaches.* Design and the Social Sciences. Ed. Jorge Frascara. London: Taylor and Francis, 2002.
19. Skibsted, Jens Martin and Rasmus Bech Hansen. *User-Led Innovation Can't Create Breakthroughs; Just Ask Apple and Ikea.* Fast Company, March 3, 2007. `http://www.fastcodesign.com/1663220/user-led-innovation-cant-create-breakthroughs-just-ask-apple-and-ikea`
20. Treviranus, Jutta. *Leveraging the Web as a Platform for Economic Inclusion.* Behavioural Sciences and the Law 32: 94-103 (2014).
21. Vanderheiden, Gregg, and Jutta Treviranus. *Creating a Global Public Inclusive Infrastructure.* Universal Access in Human-Computer Interaction Design for All and eInclusion. Berlin: Springer, 2011. pp. 517-526.
22. Wakkary, Ron. *A Participatory Design Understanding of Interaction Design.* Science of Design Workshop, CHI 2007.