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, co-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 [5], Beyer et al. [2], and IDEO [13] 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. [2]).

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 [13]), 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 [5].

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 to the overall usability of a system [23].

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 [22]. User input gathered through human-centered design methods tend to be seen only a means for users to inspire the creative process of "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 focus 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] [17]. 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 [25]. While participatory 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 do not 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 do not fully support the participation of "extreme users" — those at the margins or who have particular needs that cannot be easily accommodated by traditional user research or workshops [20]

2 Fluid's Design Tools

Taking up the co-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" [21], 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 tools such as Fluid Infusion[15], a software development framework that enables applications to be reconfigured in context- and preference-sensitive ways, and the GPII Nexus[12], which can integrate diverse software components together in a way that will eventually be supported by graphical authoring and programming tools accessible to non-developers.

2.1 Catalogue of 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 maintainers 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 [16] and the Inclusive Learning Design Handbook [14]) based on the design context and the needs of participants and project stakeholders. We outline some of Fluid's community-driven design methods below.

UX Walkthroughs are a hybrid technique based on heuristic evaluation and cognitive walkthroughs. They emphasize paired or collaborative evaluation of user interfaces by designers and non-designers alike, and serve to bring a diversity of perspectives to bear on the design process.

The UX Walkthrough technique is a procedure for critically examining a user interface by following one or more predetermined scenarios and making assessments based on common user experience heuristics such as those by Jakob

Neilsen [19]. It is an amalgam of several proven conventional inspection procedures, supporting reviewers in making assessments both from the user's point of view and that of a design expert. Pairing actual users up with designers to define scenarios and participate in walkthroughs can further enrich the results.

The multifaceted nature of the UX Walkthrough enables reviewers to make assessments across several dimensions, including: general design quality, task-oriented usability, assistive technology usability, accessibility standards compliance, and code quality. A UX Walkthrough can be performed by novices as well as experienced evaluators. The result is a comprehensive and multidimensional report of usability and accessibility issues in a website or application.

Personas and Use Cases provide models of potential stakeholders who may use a product or service and the scenarios they may encounter when using it. Personas often play an ambivalent role in an inclusive design process; as tools, they offer teams a useful way to identify with and design for certain users, yet they also simultaneously risk stereotyping or reducing users to static, product-oriented identities.

Although personas represent fictional people, their characteristics, needs, goals and motivations are rooted in the insights and feedback collected from various sources including formal or informal research techniques (such as interviews and surveys), or through familiarity with the needs and interests of self, co-workers, friends or family members. They begin as early, provisional sketches and often evolve iteratively as more information is gathered. Personas are behavioural models; they do not represent the full demographics of any given population of complex and unique people. They enable designers, developers and evaluators across a project to keep a broad and diverse collection of stakeholders in mind. Considering non-obvious or unconventional users helps a design team to think broadly and stay open to unpredicted uses of the systems they are creating. Personas are most useful for inclusive design when they are understood as full and idiosyncratic individuals, rather than as representatives of a broader category of disability, age group, or market demographic [20].

Use cases describe particular scenarios in which a persona may encounter and use a product or service, providing more detail about specific tasks and goals as well as helping to map out the potential steps in a workflow. User personas and accompanying use cases are not meant to exhaustively describe all potential stakeholders or situations; rather they help to illustrate key goals and behaviour patterns related to the design in question.

When paired with the other tools, particularly User States and Contexts and UX Walkthroughs, personas and use cases can help to paint a clearer picture of a broad and diverse range of user needs and preferences. They must be used with caution, since by their nature they create a distinction between user and designer, and they must be tempered with the awareness that no single persona or group of personas can independently determine the full range of potential uses of a product or service. Most importantly, like the real world, they need

to be understood as always in flux; user needs and goals change significantly in different contexts and at different times.

User States and Contexts 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 diverse user needs and perspectives individually or in aggregate.

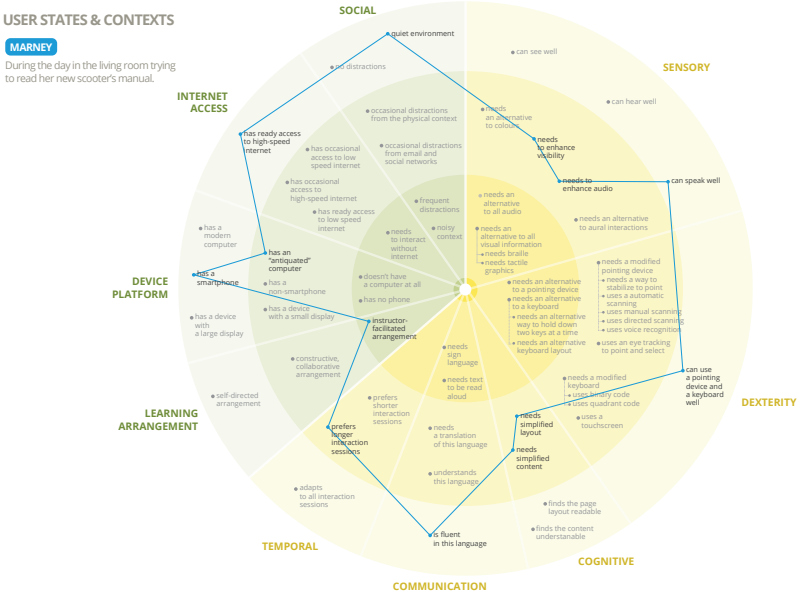

Fig. 1: A user states and contexts diagram.

These diagrams can be considered a use-modelling tool for evaluating the ability of a design to be consumed and operated by users in a wide range of states and contexts. A user states and context map can be used to demonstrate the range of needs of users that are represented by a particular persona, or those of a collection of personas. The map can also be used to consider and demonstrate how a user's state and context can change in the short term (e.g. on a daily basis) or the long term (e.g. over a lifetime). By demonstrating the many states and contexts a user can be in at any given time and in any given situation, it presents a bigger picture by describing the broad and ever-changing range of needs of a single user, and by allowing a comparison of the needs of multiple users, thereby revealing patterns or commonalities — or even outliers and edge cases that are worthy of elaboration — in needs that might not otherwise be obvious.

Community Design Crits bring together designers, developers, and users to discuss and critique design artefacts, including ideas, scenarios, mockups, and in-development software.

Design crits provide a space, both locally and remotely via the use of video conferencing tools, for informal feedback and input to be gathered on a regular basis from a group of people with differing perspectives and differing stakes in the design. Gathering feedback in this way allows everyone a chance to participate in the design process from the start and in so doing reduces the typical design/develop/test cycle time.

Open, Transparent Sharing of design artefacts and discussion on mailing lists and other community forums based on lazy consensus governance principles [3], which support community-based decision-making processes.

When working transparently, diverse participation in the design process is more likely, as those who wish to get involved and who have access to the content and tools can contribute. Accessible and open communication tools can provide both a means of alerting the community to a group discussion or other activity, as well as a means of distributing collaborative artefacts including meeting minutes, design mockups, and other relevant information.

2.2 Using and Exploring Design Tools

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 [24]. 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 is not in place to ensure that diverse contributions, especially those by non-coders, are recognized and promoted [8]. Access to open source collaborative forums, which are often 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.

3 Technological Tools for Design

Technological tools to support such open and interactive design processes are, in our experience, relatively scarce. This is not just because of the rarity of the processes they would be designed to support — rather, we argue that the underpinning for them is missing even at the technological level. In this section, we

survey the parallel story of technologies designed to support software developers in order to highlight what is missing for design, and to imagine the types of support which could be created to support a community-oriented design process.

3.1 Distributed Version Control for Software Developers

Distributed version control systems gradually took over from traditional centralised version control systems during the 2000s. Distributed versioning offers a more democratic, "peer to peer" model for managing community assets, rather than the previous authority-based model whereby the definitive version of a community asset was stored in a nominated central repository. The system with the greatest mindshare currently 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, git offers 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 be 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 is 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 is worth noting that this pull request workflow is particularly a facility of GitHub rather than simply git itself — although it is 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. However, alternatives to Git and free alternatives to GitHub exist, such as Gitorius on git, or the darcs hub[6] on darcs (although this currently has no pull request UI).

3.2 Nothing for Designers

Nothing similar to GitHub's pull request user interface and workflow currently exists to facilitate contributions to open source culture from non-developers. Because of this, open design teams are often left to find ad hoc processes, such as exchanging designs as binary files (Adobe Illustrator files, Adobe Photoshop files and PDFs) using Dropbox[7] or other cloud filesharing systems, attachments to wiki pages, or sometimes using the non-distributed SVN version management system (which scales better than git with large binary files). Comments and discussion, as a result, must be separated from the design artefacts and occur in parallel channels such as emails, IRC chats and Skype messages. Not all of this discussion can be easily archived, and even when it is, there is still a tendency for it to become separated from the design artefacts in question. This complicates lazy consensus and collaborative governance for design, and makes it difficult to establish stable, shared archives of design knowledge within a community to help bring newcomers on board.

Additionally, none of the design tools most commonly in use today have accessibility features to facilitate contributions from individuals with disabilities, such as the ability to attach descriptive text narratives to wireframes and mockup images. This huge discrepancy between the quality of tools provided to different kinds of contributors further erodes the ideology of a "meritocracy" in open source communities.

3.3 Fluid's Dream of an Inclusive Tool Chain

We dream that the same community affordances can be extended to all kinds of contributors. The technical, economic, and social barriers to this are, however, formidable. Providing, for example, an accessible equivalent to Adobe's Illustrator or similar visual design tools is a prohibitive task. Even where open source alternatives to popular commercial visual design tools exist, they are unlikely to provide useful accessibility and collaborative features that can be integrated with robust, decentralised version management in a manner suitable for non-technical users. It is clear that such tools will 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.

4 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.

4.1 Working with Design Landmarks

A important way of organising such requirements at the technical level 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 or customization after the fact by users. 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* [18]. 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**, analogous 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, with its final artefacts taking the form of structured trees with landmarks rather than binary opaque blobs [1]. By creating an infrastructure which is suitable for authoring such trees, and for casting designs both visual and architectural in terms of them, we aim to support the creation of inclusive, collaborative workflows for designers of all kinds, including users.

4.2 Requirements Beyond Frameworks

Despite our technical efforts, we recognize that collaborative, open, and participatory design communities cannot be supported solely by 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 real-world problems and their solutions or other approaches like those listed in section 2.1

5 Conclusion

The design processes and technologies we describe here aim to support new forms of participation in open source software and to ultimately provide users with the

ability to materially redesign and adapt software themselves. These goals are, needless to say, highly complex, ambitious, and challenging. We believe that such processes and tools need to be prototyped, evaluated, and refined within the context of open, collaborative communities that recognize the many and diverse contributions necessary to make highly usable software. Our approach continues to evolve and grow based on real-world experience designing and implementing software for a variety of projects, and we invite others to join our community and help creatively explore these issues with us.

References

1. Basman, Antranig, et al. *Harmonious Authorship from Different Representations.* Psychology of Programming Interest Group Annual Conference 2015. Web. http://www.ppig.org/sites/default/files/2015-PPIG-26th-Basman.pdf
2. Beyer, Hugh and Karen Holtzblatt. *Contextual Design: Defining Customer-Centered Systems.* San Francisco: Morgan Kaufmann Publishers, 1998.
3. 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.
4. Clark, Colin, et. al.: *Preferences Framework Overview* http://wiki.gpii.net/index.php/Preferences_Framework_Overview
5. Cooper, Alan and Reimann, Robert and Cronin, Dave: *About Face 3: The Essentials of Interaction Design.* Indianapolis: Wiley, 2007. pp. 80.
6. darcs hub - simple version control and collaboration http://hub.darcs.net/
7. Dropbox works the way you do https://en.wikipedia.org/wiki/Dropbox_(service)
8. Emke, Coraline Ada. *The Dehumanizing Myth of the Meritocracy.* Model View Culture (21), 2015. https://modelviewculture.com/pieces/the-dehumanizing-myth-of-the-meritocracy
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. IDEO. "Methods." *Design Kit.* http://www.designkit.org/methods
14. The FLOE Project, *The Inclusive Learning Design Handbook* http://handbook.floeproject.org/
15. Fluid Project: *Fluid Infusion combines JavaScript, CSS, HTML and user-centered design*: http://fluidproject.org/products/infusion/
16. Fluid Project. *The Fluid Design Handbook.* https://wiki.fluidproject.org/display/fluid/Design+Handbook
17. Mace, R. et al *The Principles Of Universal Design* https://www.ncsu.edu/ncsu/design/cud/about_ud/udprinciplestext.htm
18. Mozilla Developer Network - Getting Started with CSS Selectors https://developer.mozilla.org/en/docs/Web/Guide/CSS/Getting_started/Selectors
19. Neilsen, Jakob. *10 Usability Heuristics for User Interface Design.* Neilsen Norman Group https://www.nngroup.com/articles/ten-usability-heuristics/

20. Pullin, Graham, and Alan Newell. *Focussing on Extra-ordinary Users.* Universal Access in Human Computer Interaction. Coping with Diversity. Berlin: Springer, 2007. 253-262.
21. Sanders, Elizabeth B.N. *From User-Centered to Participatory Design Approaches.* Design and the Social Sciences. Ed. Jorge Frascara. London: Taylor and Francis, 2002.
22. 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`
23. Treviranus, Jutta. *Leveraging the Web as a Platform for Economic Inclusion.* Behavioural Sciences and the Law 32: 94-103 (2014).
24. 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.
25. Wakkary, Ron. *A Participatory Design Understanding of Interaction Design.* Science of Design Workshop, CHI 2007.