You say tomato, I say tomato, let's *not* call the whole thing off: the challenge of user experience design in distributed learning environments

Jutta Treviranus

Jutta Treviranus is a Director based at Adaptive Technology Resource Centre (ATRC), University of Toronto, Toronto, Canada.

Abstract

Purpose – *The purpose of this paper is to chronicle new user experience (UX) design approaches being pioneered in an international, multi-institution, multi-sector, cross-project initiative called the Fluid Project, covering the strengths and shortcomings of these approaches and the lessons learned about design and development in distributed communities.*

Design/methodology/approach – *Open source and community source software development projects have not fulfilled their promise of innovation and natively optimized tools and applications in large part due to a lack of integrated UX design and development processes. Fluid has developed a UX approach that aims to address the need to accommodate the huge diversity of users and contexts in academic communities as well as the critical need to improve the user experience.*

Findings – *It has been found that the Fluid approach challenges common or traditional notions integral to teaching in higher education, software design, user interaction design methods, usability research and accessibility strategies. It proposes greater individual control over the UX than most users may be ready to assume despite obvious benefits. An unexpected UX challenge is creating tools and applications that prompt and support users in configuring their systems to their personal needs and contexts.*

Originality/value – *Fluid has designed and prototyped new UX design methods, pedagogical practices, and usability and accessibility approaches to suit the context of distributed academic communities and open source development, while at the same time producing a UX system of benefit to the mashup or integration of any set of disparate tools.*

Keywords *Design and development, Higher education*

Paper type *Research paper*

The community source movement within higher education has dramatically changed the dynamics of software and information technology practices within academia. Institutions of higher education have regained control of the tools that are critical to their mission and of the intellectual property contained in these tools by collaboratively establishing distributed development teams to collectively develop software systems. Despite the unarguable success of this movement, in several senses the community source projects have not lived up to their promise. Many universities and colleges committed to the initiative in hopes of helping to create tools that are better tailored to the practices, processes, values and culture of their institution. These institutions envisioned tools that not only provided a viable alternative to commercial counterparts but offered platforms for innovation in academia. While the features and functions of the software systems collectively developed have progressed well towards these goals, the user interface or user experience of these systems has not. Poor or inconsistent design and development of the user interface (UI) is widely recognized as a systemic problem within academic community source or open source projects and the major impediment to more widespread adoption (Nichols and

Twidale, 2003). This problem is also a barrier to innovation - to improving learning, research and administration activities within academic software. There is widespread agreement that the greatest need for innovation in this field is in the area of human interaction and support for more effective academic workflow. However, cumbersome, problematic UI development processes and UI frameworks within community source software projects make it very difficult to contribute innovations (Thomas, 2002).

For a number of reasons UI development in these projects is commonly left to programmers, with little input from skilled designers (who appear to be in short supply). It is frequently tackled at the end of the development process. Components of the UI are often developed redundantly, inconsistent across applications and inadequately tested and refined. Architectural frameworks for the UI are also inconsistent, redundant and poorly thought out. It is speculated that while project members share a strong sense of vision and commitment to intended functionality, they do not have the same shared vision regarding the intended interface (Nichols and Twidale, 2003).

Another challenge faced by community and open source software projects in academia is that they must address the needs and preferences of a very diverse group of users and constituents. These differences arise from institutional preferences (including branding); conventions of an academic discipline (e.g., math vs. English); cultural or linguistic differences; differences related to age, role or perspective; different teaching and learning approaches; and differences related to disability and environmental constraints. At the same time it is widely acknowledged that a consistent UI across tools and functions would greatly improve the user experience (especially when using an ever-increasing set of tools) and predictable, familiar interaction conventions are prerequisites for ease of use. However given the diverse community in most open or community source project, it is very difficult if not impossible to agree upon a single UI design and when a single UI design is chosen many community members feel disenfranchised. The need and difficulty intensify as the interest in integrating academic software systems increases, further necessitating cross-project UI coherence. Current trends and motivations that push toward the creation of modular and distributed learning environments and curriculum to meet localized needs also increase the challenge while intensifying the need for a consistent UI.

The Fluid Project (2008) attempts to address these challenges by providing a UI architecture that supports UI transformation at runtime for individual needs and preferences or during configuration for institutional preferences. This transformation is made possible by a rich, living library of reusable UI components that have been tested for usability and accessibility. The architecture and UI components work across Web platforms and applications. Thus each user can have a consistent, personally-optimized user experience across tools. At the same time the software project or the implementing institution can meet the diversity of needs of its users without needing to prioritize one community over another or compromise the design of one group for the needs of another. In addition Fluid provides a toolkit of user experience design resources to help in optimizing the user experience during the design, development, configuration and implementation of the applications. Fluid is an international project funded by The Andrew W. Mellon Foundation in its second year (http://fluidproject.org). Participating software projects include Sakai, Kuali Student, uPortal, Moodle, ATutor, CollectionSpace, OpenCast and others.

The Fluid Project (2008) is structured such that Fluid project members are embedded in, recruited from and active participants in a number of community and open source software projects with the goal of not only improving the user experience or user interaction design of the applications produced by the projects but also improving the attitudes, knowledge, collaboration, and design and development process within the communities. In more mature projects, polarized groupings had formed between coders or developers and community members concerned about usability, quality assurance, accessibility and user experience design. The established value system within these software projects, which assigned credit according to the amount of code committed, did not adequately recognize contributions from non-coders such as designers, instructional designers or usability experts. Whether as

cause or effect, very few designers were active participants in these communities. The design and development process did not make room for design/accessibility/usability input until the very end as a critique, evaluation or gate-keeping process. Consequently, any changes suggested required costly retrofits; and, coming from individuals who were not integrated or trusted members of the team, were frequently received defensively. At worst evaluation outcomes were minimized or rejected citing lack of time and resources needed to address the problem and the greater importance of releasing a product. Designers, relegated to the role of evaluators, were labeled as ''whiners'' or discontents, to dismiss their input. Software project participation in Fluid entailed a commitment by the community to address these structural impediments to good user experience (UX) design by considering UX design in governance structures and the organization of development processes.

Surprisingly, these anticipated process challenges, thanks to the sophistication and self-awareness of the software project communities participating, while not simple, are being addressed with the collaboration and cooperation of the previously polarized groups. Once the issues and potential solutions were identified, most of the communities took a pragmatic and determined approach to addressing the issues. An informal analysis of the community forum postings shows a significant cultural shift, with much greater value and respect given to user experience design input. Community processes and governance structures have also made room for user experience design throughout the development cycle of the applications.

However, one of the unexpected outcomes of Fluid thus far is that we have found ourselves at odds with common or traditional notions integral to pedagogy, software design, user interaction design, usability research and accessibility strategies. In addition, while no user will deny that they prefer applications tailored to their needs, few users demand or take advantage of personalization functions. Consequently Fluid must design and prototype not only new UX design methods, teaching supports, usability and accessibility research methods but also systems that invite and support the user to adjust the UX to meet their individual needs and contexts.

The user interface as contested real estate

The Fluid approach makes it possible to personalize the application to your individual needs and preferences. When fully implemented, each user should be able to create a portable personal preference profile that they can take from application to application (based on the ISO 24751 ''AccessForAll'' standard, a standard that defines a common language for expressing personal preferences and corresponding metadata to label resources to match these personal preferences) (www.imsglobal.org/accessibility/). Each application would automatically approximate the specified user interface configuration. The technical challenges of this approach are many but surmountable. An unexpected challenge has been the reluctance of users to customize their user experience despite the obvious benefits of this approach for both addressing personal needs and preferences and creating a consistent user experience across applications.

While the software tools being created are akin to our offices, desks, studios, meeting spaces or workshops and take on the role of tools and instruments critical to our profession, craft or learning experience, users do not view the applications as tools or environments that they can personalize. While we take great care in organizing, customizing, decorating and arranging our desks, offices, workshops, meeting rooms and toolkits to match our personal needs, preferences and tastes, we do not generally feel the need or license to customize our applications to our personal needs (see, http://blogs.msdn.com/jensenh/archive/2006/06/27/648269.aspx, a Microsoft study showing that only 2 percent of users customized their word processing interface, although more recent studies show that there is a trend toward greater use of personalization). Consider the highly personalized and specific arrangements of an artist's palette, a carpenter's workbench, a writer's desktop or a teacher's classroom, each of which is no more critical to the work or profession they support. In contrast the computer desktop and applications have become contested real estate, controlled by a

number of interests other than the interests of the users of the tools or inhabitants of the virtual environments.

A growing portion of that real estate is being claimed by advertisers and commercial interests, whether the content is used to brand the product itself or completely unrelated products that can only act as a distraction from the task at hand consciously or subliminally. We seem puzzlingly forgiving of the clutter caused and space wasted by this appropriation of our virtual desktop, especially when using Web applications.

In academic software applications it is frequently the academic institutions that use large portions of the available workspace to brand the institution and possibly also the faculty and the department. It is questionable whether educators, students or administrators require the use of limited desktop space to remind them of their institutional allegiance to successfully complete the frequently complex tasks to be performed with the software application.

In some cases we have given our personal tools and spaces over to developers who view the UI design as a trivial addition to the important work of creating the functionality of the application and colloquially refer to it as ''lipstick''. (The term chosen, and the manner in which it is characteristically uttered, raises interesting questions regarding the gender associations with the fields of UX design versus development.) The arrangement of the menu, the toolbar, the prioritization of the tools, the color scheme, the choice of font, the mental metaphors used and many other non-trivial details are left to individuals who give these characteristics little regard.

Increasingly we give in to external choices made on our behalf regarding security. We let the platform, successive applications, systems and environments erect fences, gates and continuous disruptions to our freedom of movement and interruptions to our workflow without so much as retaining control or choice of what we want secured, from whom and when.

Of greater relevance to the Fluid initiative, we have frequently handed control of these highly personal spaces and toolkits to designers who view the UI as a mass market generic environment that should accommodate the ''typical'' user and we have adapted or shoe-horned our work habits, customs, preferences and comforts to fit this conception of the ''typical'' user, even when we acknowledge that we have unique, personal needs and are far from typical.

At present the application UI is anything but a personal environment that we make ourselves at home in or customize to our personal needs, workflow or work habits. For Fluid to achieve its larger vision it must encourage users to treat the UI as their own space. Thus a critical Fluid design challenge is to design a user experience and user experience components that entice, encourage and make users comfortable with ''fiddling with'' or customizing their application UI.

Given that users are reluctant to customize the UI configuration themselves, one approach may be to make intelligent inferences and adapt the interface for the user. Preliminary studies have shown that users do not welcome this adaptation on their behalf unless it is completely accurate in its prediction or the user is informed or asked for confirmation of the inferred preference. This ''intelligent'' adaptation must therefore be done sparingly, carefully and with full transparency and reversibility.

One barrier to personalization has been the risks associated with adjusting the interface. Microsoft cites the numerous unintentionally reconfigured interfaces. To prevent this unwanted situation, any adjustment must have an easy way of resetting or undoing the requested changes. Users must also be able to preview the full effect of their configuration choices before committing to them.

The arguments for personalization are equally relevant to collaborative spaces. Next time you attend a meeting, note the determined way in which participants position themselves in the room or in relation to others, equip themselves with tools and emblems, adjust the shared materials, influence the order of events and perform other rituals and routines to configure

the shared environment to their needs and the requirements of the shared activity. Personalization or individual customization has its place in collaborative spaces, it is simply complicated by negotiation and compromise when the personal and collective needs differ. The equivalent is rarely seen in forums, chats or online meeting environments and yet the motivations for reconfiguration are just as compelling.

The personalization facilitated through Fluid should be distinguished from personal portal pages such as iGoogle and myYahoo which customize content presented according to the inferred interests of the user (and devote a great deal of real estate to branding). It can be argued that this form of personalization is counter to goals of inclusion and diversity. It does not encourage users to consider alternative notions, new ideas, divergent thinking or marginal views.

Agile, participatory, open source UX design

The Fluid approach to user experience design, development and usability testing is also at odds with standard or commercial UI design methods. While UX or UI designers are rare in community or open source environments, most skilled designers and usability experts that do participate are familiar with a corporate, commercial design process based on methods promoted by Cooper (2008) and others (www.cooper.com/about/process/research.html). These are incompatible with:

- the success criteria of the Fluid approach;
- the distributed nature of the community;
- the agile development process;
- the resource realities of open and community source projects; and
- the role and nature of users of the applications being developed.

Even ''user-oriented'' or ''user-centered'' methods generally assume that the user really does not know what is best or what they want. The common assumption is that users are not self-aware, what they report doing is not actually what they do and asking users what they might want does not lead to innovation because they extrapolate from what they know and are most likely to ask for the equivalent of a faster horse carriage than a car. Consequently the assumption is that any proposed design requires extensive user research with observation by objective expert UI researchers and data gathering from a large number of representative users. Part of this assumption is that reliable information can only be obtained by observing many users in controlled situations for each feature, function or experience to be designed or evaluated, not by asking the user or a group of users (Virzi, 1992). There are several problems with this assumption and approach in community and open source academic software projects especially when applied to the reconfigurable UX developed through Fluid:

- users are too diverse to get a representative set for testing;
- the contexts are too diverse and developers, designers and users are too distributed to achieve the critical mass needed to statistically verify research findings;
- in Fluid there is no single configuration of the interface to test as each interface is configured to the individual needs of each user; and
- users within community and open source are much more self-aware and articulate in the methods of UI design and are not content to be studied but want to be active participants in the design.

In addition to these problems with traditional commercial UX design methods, the criteria for a ''good'' or ''usable'' design are different, more complex and deeper. In some ways usability testing must be far more rigorous and in other ways it can be much more forgiving. The UI design must be amenable to diverse configurations. The UI must be highly flexible. It must gracefully survive translation to alternative modalities and languages (e.g., from audio to

visual, from text to audio, from English to Mandarin). It must support users in the many dimensions of diversity including the novice and master or skilled users. Visual design is not as important. Conversely, the designer does not need to commit to a single design, configuration, behavior or workflow. This makes room for risk-taking, playfulness and creativity in the designs.

Consequently Fluid moved toward a largely qualitative, action research, participatory design approach to UX research at both the design and evaluation stage. The questions to be addressed by this research include: is there a large enough range of choices to accommodate the range of potential users, and is the method of adjusting the interface easy to use and inviting of adjustment? Consequently there is little concern with garnering a ''clean sample'', or adequately controlling or replicating the conditions across trials. The concern is that testing includes sufficient variety in users and conditions of use and that the design survives across each of these diverse trial conditions while achieving usability and accessibility criteria. Thus, successful UX research within Fluid more closely approximates software testing in that it is dependent not on controlling variables to achieve statistically reliable results but on testing as many variations in contexts, uses and configurations as possible to make sure that the fundamental design survives across these conditions.

The community and open source environment is participatory by nature, lending itself well to participatory design practices. Agile programming with frequent small functional modular builds and explicit, public, transparent communication (as the side effect of a distributed work environment) set the stage for broad participation in the design. By creating mechanisms for providing input and by sharing designs as well as code through mock ups, wire frames, design patterns, scenarios, use cases and other means enables informal participation in the design.

Further chunking design problems into manageable modules, providing easy to follow testing protocols that are tolerant of contextual variation and report templates that capture the necessary information needed to interpret the variations, make it possible to distribute usability and accessibility testing and design input across a number of locations and contexts. As previously stated, given the design criteria of transformability and integrity across configurations; the more diverse and numerous these testing conditions are the better.

The Fluid UX research goal of the most varied and representative user input is assisted by the VULab project. VULab enables remote user testing by recording a video of a remote user's on-screen interactions with a Web site including any verbal user comments. It incorporates a survey tool and does not require the installation of software on the user's computer. Data is stored on a central server for analysis (Owston and Edwards, 2008, http://wiki.fluidproject.org/display/fluid/VULab + Project + Scope#VULabProjectScope-ProjectDescription).

An agile design method is best suited to the agile Fluid UX development process. Given that the user experience reaches deeper into the functional architecture than usual to enable transformation, proceeding with development requires a certain level of user experience design. This must be agile and iterative to match development. Several authors have argued that robust designs can be created based on a surprisingly small amount of user data (Nielsen, 1993). This is even more the case when the user experience is highly configurable. However, there are additional requirements of testing: to determine the scope of the transformation, to enumerate the range of workflows and to ensure that the necessary interactions upon which to build the possible configurations are adequately captured.

The Fluid UX approach is highly dependent on a core architecture that supports and sustains this flexibility and variety. Given the instability and unpredictability of the UI, the design and development of the entire system requires greater cognizance of what is to be achieved, in what order, in what context, given what priority or success criteria – rather than how to achieve it. The core architecture must also flexibly glue together a variety of components and their configurations. This requires much closer communication between

the design and development teams within Fluid, requiring more compatible design and development processes.

The issue of innovation and the limits to the imagination of the average user are no more addressed through more commercial UX design methods than through their alternatives. The flexibility of the Fluid UX makes room for experimentation, enables the survival of extremes and designs that ''break the mold''. The quirky or unusual design need not be rejected as it can be easily reconfigured, transformed or replaced at little or no cost and does not take the place or disadvantage more conventional designs. Because Fluid design does not need to cater to the norm, the typical or the popular, there is far greater opportunity to take risks and pursue unique ideas and give serendipity free reign. This invites playfulness in designers, developers and users. This in turn is more likely to lead to more broadly implemented innovation.

Inclusive design: rethinking accessibility

One of the most critical goals of the Fluid project is to address accessibility for students, instructors and administrators with disabilities. Most academic software applications fail to meet legislative accessibility requirements in their respective jurisdictions. Most academic institutions implementing these systems are at risk of litigation. More importantly the ever-increasing number of students, educators and administrators with disabilities are excluded from using these systems and are thereby denied access to education or vocation.

Aware of the risks and inequities, members within each community are frantically addressing the most serious accessibility issues with each release using a ''band-aid'' approach. There is no system-wide strategy for accessible design, consequently as the toolset grows the effort required increases. Because of the presentation frameworks and architectures presently employed, each new contribution of code brings with it new accessibility issues. Developers frequently find the accessibility guidelines constraining and restrictive of creativity and innovation. This causes tension between accessibility advocates and more creative developers within the community.

If the present approach to accessibility continues and is successful, at best, the minimum criteria for accessibility will have been met and no user with a disability will find it impossible to use the system. However, while the system may be minimally accessible to all, it will be optimized for no one.

Fluid enables the customization of the user interface based on the accessibility needs and preferences of each user. Thus the user experience of one student need not be compromised to accommodate the accessibility needs of another student. The system can be optimized for each individual user's access requirements. Not only will this greatly benefit users without disabilities, it will provide usable interfaces for users with disabilities whose needs and preferences differ greatly and cannot be addressed through a single set of guidelines. From a development perspective, application developers need not be concerned with the details of accessibility guidelines as long as their applications are compatible with the Fluid components and architecture, accessibility can be delegated to the UI components. User interface designers can explore creative new approaches and technologies without worrying about breaking accessibility legislation. UI developers with accessibility expertise can create robust UI components that can be repurposed with a number of tools across multiple iterations. Because each user interface component can be specialized, creative new UI designs can be explored to meet a greater diversity of accessibility needs.

The Fluid approach challenges the currently legislated approach to accessibility which requires a rule-based, one-size-fits-all solution (www.w3.org/WAI/Policy/). As where Fluid provides accessibility through a flexible system that adjusts to the individual needs of each user, the traditional approach is to adhere to a set of fixed accessibility rules for every resource or resource configuration.

The Fluid project frames disability as a mismatch between the needs and preferences of the user and the system or environment provided (Treviranus and Roberts, 2006). In this framing disability is not a personal trait but the result of a system that does not adequately address user needs. Thus an individual who is blind would not have a disability in a spoken lecture, but someone who does not have access to or knowledge of the background material would be. Accessibility is therefore determined by the capability of the system to adapt to the needs of each user. Given the Fluid approach, no resource or application can be categorically labeled as ''inaccessible'', or ''accessible'' for that matter. Resources, applications or user interfaces match the needs of a given set of users. Resources or interfaces that are sufficiently flexible to meet a large range of needs can be said to be more accessible.

Fluid implements ISO 24751, also referred to as AccessForAll, to help achieve this match. This is a multipart standard that provides a common language for describing each individual's user interface preferences and needs in a given context in one part; and, a common language for describing resources, including user interface components, such that they can be matched to user needs in another part (www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber = 41521). This requires compliance testing of a system and process rather than of each individual resource or Web page. The AccessForAll approach is also less specific and more encompassing in who is to be served by more accessible systems and does not adhere to classification schemes commonly applied to people with disabilities for the purposes of policy or legislation. AccessForAll classifies only at the functional level; i.e. require alternative but equivalent:

- display;
- control; and/or
- additional support.

Rather than grouping users, each user can represent several different configurations associated with the different contexts in which they use the system. The Fluid community is working to update accessibility policy and legislation to support this approach.

Relinquishing control of the presentation

One tenacious characteristic of the designer, educator and user, that is incongruent with the Fluid approach to user experience and accessibility, is the obsession with the visual presentation. In cognitive walkthroughs with educators and educational content producers, one of the first concerns or preoccupations is with the presentation of the material: what visual theme will be used; will the focus be images, text, animation, audio or video; how will the transitions occur, etc. Consequently these presentation characteristics are ''hard-coded'' into the curriculum, either technically or conceptually. This makes it difficult or impossible for the learner to adjust the presentation for reasons of preference, learning style or accessibility. This need to control the presentation is also inconsistent with ever more popular Web 2.0 based learning experiences collaboratively constructed by learners.

The accessibility approach supported by Fluid works best if the content producers focus on the educational goal and the narrative or structure of the content (both temporal and semantic, e.g. what do I need to communicate first, what learning outcome do I want, what concept do I build upon this last concept and so on) and leave the presentation as flexible as possible. This requires content authoring tools that support this approach. This is one of the goals of an accompanying project called TransformAble (ATRC (2008) http://transformable.atrc.utoronto.ca/). TransformAble consists of a number of Web services that collectively transform the user experience within a Web application to match the needs or preferences of the user. Among the services are a preference service that enables the user to set portable preference files for specific contexts, a styling service that creates the needed styling mechanism (e.g., CSS or XSLT) to restyle the user interface, a content aggregator that finds, retrieves and displays alternative content (e.g., captions for audio and descriptions for video, alternatives to simulations, etc.) and an authoring tool that supports educators or

content authors in creating content that is amenable to transformation and the metadata needed to match the content to user needs.

Designed for heterogeneous distributed communities

The distributed UX design research methods of Fluid are well suited to open and community source projects. No one company could achieve the necessary diversity of users and contexts. Similarly the Fluid approach to UX development and the individual configuration of the application interface can only be achieved when many players with different strengths and perspectives pool resources. This UX design approach facilitates a more inclusive community in that no contribution is without utility as long as it follows the agreed upon open standards, this in turn encourages broader participation.

The development and design must also be largely open source or at least fully open standards compliant to achieve the necessary interoperability and cooperative behavior of the applications needed to transform and reconfigure the applications to the needs of each user or context of use. Conversely, (as has been the experience in academic community source projects) tight coordination, agreement and communication of monolithic, homogeneous configurations are not well suited to distributed communities. The Fluid methods and technologies capitalize on diversity and variety inherent in cooperative distributed communities. At the same time the Fluid approach is beneficial to any heterogonous, distributed learning environment in which disparate widgets, modules or tools are integrated; whether open source or proprietary, or a combination of both.

To achieve the type of personalization envisioned by Fluid requires a host of talents, perspectives and resources. However, at the same time, Fluid affords software projects and participating institutions greater development efficiencies in that developers can focus on developing new and innovative functionality while relying on the Fluid component library, architecture and UX toolkit to achieve a usable, accessible, innovative and robust user interface.

Conclusion

Fluid is a bold experiment in user experience design with many issues yet to be addressed but with the promise to enable unprecedented innovation and critical inclusion. It enables effective participation by a diversity of contributors. It offers a sustainable, integrated approach to accessibility. It applies much needed talent and concerted effort to the UX of community and open source software projects. It creates a system infrastructure that capitalizes on distributed collaborative communities. It has also become a testing ground for novel approaches to UX design methods, usability testing and the delivery of education in an online system. Lastly, it offers an environment in which to explore the user relationship with the tools and applications that have become integral to our daily life.

References

ATRC (2008), ''Project TransformAble'', available at: http://transformable.atrc.utoronto.ca/ (accessed 4 November 2008).

Cooper (2008), ''Research'', available at: www.cooper.com/about/process/research.html (accessed 4 November 2008).

Fluid Project (2008), ''What is fluid?', available at: http://fluidproject.org (accessed 4 November 2008).

Nichols, M.D. and Twidale, B.M. (2003), ''The usability of open source software'', *First Monday*, Vol. 8 No. 1, available at: http://firstmonday.org/article/view/1018/939 (accessed 4 November 2008).

Nielsen, J. (1993), *Usability Engineering*, Academic Press/AP Professional, Cambridge, MA.

Owston, R. and Edwards, B. (2008), ''Open Virtual Usability Lab'', v.34, available at: http://wiki.fluidproject.org/display/fluid/Open + Virtual + Usability + Lab (accessed 4 November 2008).

Thomas, M. (2002), ''Why free software usability tends to suck, and how to improve it'', available at: http://mpt.net.nz/archive/2008/08/01/free-software-usability (accessed 4 November 2008).

Treviranus, J. and Roberts, V. (2006), ''Inclusive e-learning'', in Weiss, J., Nolan, J. and Trifonas, P. (Eds), *International Handbook of Virtual Learning Environment*, Kluwer, Springer, Hamburg, pp. 439-65.

Virzi, R.A. (1992), '''Refining the test phase of usability evaluation: how many subjects is enough?'', *Human Factors*, Vol. 34 No. 4, pp. 457-68.

Further reading

Harris, J. (2006), ''Let's talk about customization'', available at: http://blogs.msdn.com/jensenh/archive/2006/06/27/648269.aspx (accessed 4 November 2008).

IMS Global Learning Consortium (2006), *IMS AccessForAll Metadata Specification*, available at: www.imsglobal.org/accessibility/ (accessed 4 November 2008).

ISO/IEC (2008), ''ISO/IEC 24751-1:2008'', available at: www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber = 41521 (accessed 4 November 2008).

W3C (2006), ''Policies relating to web accessibility'', available at: www.w3.org/WAI/Policy/ (accessed 4 November 2008).

Corresponding author

Jutta Treviranus can be contacted at: jutta.treviranus@utoronto.ca